Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Bachelor Thesis

Flow Visualization of Aortic Dissections

Author:

Aaron Schroeder

May 10, 2021

Advisors:

Prof. Dr. Bernhard Preim
Department of Simulation and Graphics
Otto-von-Guericke University Magdeburg

Dr. Gabriel Mistelbauer
Department of Simulation and Graphics
Otto-von-Guericke University Magdeburg

Schroeder, Aaron:
Flow Visualization of Aortic Dissections
Bachelor Thesis, Otto-von-Guericke University Magdeburg, 2021.

Statement of Authorship

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning. I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, May 17, 2021 .....

Abstract

Aortic dissection is a life-threatening cardiovascular disease, caused by blood entering the media layer of the aortic vessel wall. This splits the initial true lumen of a healthy aorta into two channels, a true lumen and a so-called false lumen. This significantly weakens the vessel wall, which may lead to fatal aortic rupture. Conventional magnetic resonance imaging (MRI) scans of a patients aorta only reveal morphological features of the dissection. While 4D flow MRI is able to capture hemodynamic data, it is limited in terms of accuracy and resolution and does not give any indication of pressure. In order to produce more detailed flow data, simulations can be performed using the model of a patient's aorta captured with MRI. Computational fluid dynamic (CFD) simulations are able to simulate hemodynamic flow very accurately, while also providing values for pressure and wall shear stress. The information obtained by such simulations can then be visualized. To explore the applicability of different visualization techniques in respect to aortic dissections, several methods were implemented and tested in this thesis. The main flow representation techniques were selected to be streamlines and animated arrow glyphs. To generate streamlines, an intuitive seed point selection method was created. In oder to seed lines evenly distributed over the cross-section of the vessel, the vessel is voxelized, skeletonized and subsequently the centerlines of the separate lumina extracted. Seed planes are placed perpendicular to the centerline and the cross-section of a lumen evenly filled with seed points. Streamlines are then generated using 4th order Runge-Kutta integration and rendered using depth-dependent halos, to reduce visual clutter. To assess the flow in both lumina and compare their flow features, we compute streamlines in both lumina simultaneously and differentiate them using textures. Lines can also be shown with a color scale, giving additional information about flow velocity or direction. By color-coding streamlines according to velocity, we can identify areas of interest. To differentiate retrograde flow from normal flow, which may be of clinical importance, we color streamlines according to their flow direction compared to the lumens centerline. Animated arrow glyphs produce a very intuitive visualization, allowing simultaneous display of flow paths, velocity, and direction, while flow features like vortices may not be recognized as accurately as with streamlines.

Kurzfassung

Die Aortendissektion ist eine lebensbedrohliche Herz-Kreislauf-Erkrankung, die durch das Eindringen von Blut in die Media-Schicht der Gefäßwand der Aorta verursacht wird. Dadurch wird das ursprüngliche echte Lumen einer gesunden Aorta in zwei Kanäle geteilt, ein echtes Lumen und ein sogenanntes falsches Lumen. Dadurch wird die Gefäßwand deutlich geschwächt, was zum reißen der Aorta führen kann. Konventionelle Magnetresonanztomographie (MRT) Scans der Aorta eines Patienten zeigen nur morphologische Merkmale der Dissektion. Das 4D-Flow-MRT kann zwar hämodynamische Daten erfassen, ist aber in Bezug auf Genauigkeit und Auflösung begrenzt und gibt keinen Hinweis auf den Druck, der in verschiedenen Teilen des Gefäßes herrscht. Um detailliertere Strömungsdaten zu erhalten, können Simulationen unter Verwendung des mit der MRT erfassten Modells der Aorta eines Patienten durchgeführt werden. Blutflusssimulationen sind in der Lage, die hämodynamische Strömung sehr genau zu simulieren und gleichzeitig Werte für Druck und Wandschubspannung zu liefern. Diese Informationen können dann visualisiert werden. In dieser Arbeit wurden mehrere Methoden implementiert und getestet, um verschiedene Visualisierungstechniken in Bezug auf Aortendissektionen zu untersuchen. Als Haupttechniken zur Strömungsdarstellung wurden Stromlinien und animierte Pfeilglyphen ausgewählt. Um Stromlinien zu generieren, wurde eine Methode zur Auswahl von Startpunkten entwickelt. Damit Stromlinien gleichmäßig über den Gefäßquerschnitt verteilt werden können, wird das Gefäß voxelisiert, skelettiert und anschließend die Mittellinien der einzelnen Lumina extrahiert. Startebenen werden senkrecht zur Mittellinie platziert und der Querschnitt eines Lumens gleichmäßig mit Startpunkten gefüllt. Die Stromlinien werden dann mit Runge-Kutta-Integration 4. Ordnung generiert und mit tiefenabhängigen Halos dargestellt. Um die Strömung in beiden Lumina zu beurteilen und ihre Strömungseigenschaften zu vergleichen, werden Stromlinien für beiden Lumina berechnet, gleichzeitig angezeigt und mit Hilfe von Texturen differenziert. Die Stromlinien können auch mit einer Farbskala dargestellt werden, was zusätzliche Informationen über die Strömungsgeschwindigkeit und -richtung liefert. Durch die Farbkodierung der Stromlinien entsprechend der Geschwindigkeit können wir Bereiche, die von Interesse sind, identifizieren. Um rückläufigen Fluss von normalem Fluss zu unterscheiden, färben wir Linien abhängig vom Winkel zwischen ihrer Strömungsrichtung und der Mittellinie des Lumens. Animierte Pfeilglyphen erzeugen eine sehr intuitive Visualisierung und ermöglichen die gleichzeitige Anzeige von Strömungspfaden, Geschwindigkeit und Richtung, während Strömungsmerkmale wie Wirbel möglicherweise nicht so genau erkannt werden können, wie bei Stromlinien.

Acknowledgments

We thank Kathrin Bäumler (3D and Quantitative Imaging Laboratory, Department of Radiology, Stanford University) for providing flow data of aortic dissections.

Contents

List of Figures

1. Introduction

Aortic dissections are a rare but life-threatening cardiovascular disease [5], characterized by the separation of the vessel wall layers, specifically the tunica media by ingress of blood. The development of a second flow channel, called false lumen, with varying proximal and distal extension is caused by an entry tear in the tunica intima. True and false lumen are separated by a so-called dissection flap or dissection membrane, resulting in a weakened outer wall of the false lumen and possibly fatal aortic rupture. Further possible results of aortic dissections are pericardial tamponade and malperfusion of branch vessels, some of which require urgent surgical or endovascular treatment [6].

The most frequently used classification for aortic dissections, the Stanford classification, distinguishes dissections in regards to the entry tear location. Type A dissections involve the ascending aorta and are 2-3 times more common than type B aortic dissections, which only involve the descending aorta [7]. To detect and treat later complications of aortic dissections, including false lumen dilation and aneurysm formation, survivors are dependent on clinical and imaging monitoring for the rest of their lives [6].

Although classification, treatment and prognosis of aortic dissections are strongly informed by morphological features captured on imaging studies [8], phantom studies suggest, that true lumen collapse and branch vessel malperfusion are related to number, size and location of intimal tears as well as the distribution of branch vessels draining the false and true lumen [9; 10]. Increased false lumen pressure possibly promotes false lumen dilation and subsequent aortic rupture [11; 12]. Limited false lumen outflow showed to contribute to overall disease progression [13; 14; 15; 16]. Noninvasive hemodynamic measurements, using 4D flow magnetic resonance imaging (MRI), provide insight into functional features of a patient's vascular system. In consequence of special instrument requirements (High field strength MRI) and high cost, these scans are not routinely obtained. Alternatively, computational fluid dynamic (CFD) simulations emerge as an increasingly reliable and more accessible way to gain patient-specific, detailed hemodynamic data [17; 18].

Better understanding and more accurate interpretation of hemodynamics in aortic dissections can help timing and selection of medical, surgical and endovascular treatment. Obtaining detailed functional data aids the development of new treatment approaches and the ultimate improvement of patient quality of life, but the accurate and clear visual representation of complex hemodynamic data is important for the correct interpretation and treatment selection.

Patients suffering from thoracic aortic dissections have a 10-year survival chance of 35% to 75% [19]. However, the outcome varies significantly between patients with complicated dissections and those with uncomplicated dissections. Type A aortic dissections require immediate surgical repair, whereas uncomplicated Type B dissections can initially be treated medically most of the time [20]. Unfortunately, up to 30% of patients with Type B dissections, that can initially be treated medically, eventually require an intervention [21]. The ability to predict, which patients will eventually need surgical treatment, may allow earlier treatment to avoid irreversible organ ischemia or fatal aortic rupture [19].

1.1 4D Flow MRI

Prognoses for the development of aortic dissection still primarily relied on justified but uncertain assumptions about the hemodynamics of the aorta, which are based on morphological features. Using 4D phase contrast MRI, the flow inside the aorta can be directly measured and evaluated. Stroke volume, primary entry tears, helical flow, and velocity were found to be related to the rate of aortic expansion [22]. Pressure differences between true and false lumina, that lead to flow obstruction of the true lumen by the dissection flap, may also be identified using 4D flow MRI [19].

1.2 Computational Fluid Dynamic Simulations

As 4D MRIs are able to capture fluid velocity only, simulations have to be utilized to gain different information, which are possibly clinically important. Computational fluid dynamic (CFD) simulations for aortic dissections can be performed using patient-specific computed tomography (CT) and MRI scans. Simulations with rigid wall assumptions were able to show a correlation of elevated false lumen pressure and subsequent wall shear stress with disease progression [23; 24; 25]. Simulations of thoracic endovascular aortic repair also showed increased false lumen pressure with restricted outflow and a reduction in false lumen pressure with restricted inflow [26; 27]. Studies aiming to validate CFD simulations against 4D flow MRI [28] and 2D flow MRI [29; 30] found errors of 20% or more between measured and simulated flow data. Reasons for these errors could be rigid wall assumptions used in the investigated simulations and measurement inaccuracies.

To more accurately simulate hemodynamics inside aortic dissections, dynamic deformation of vessel wall and, most importantly, the dissection flap have to be incorporated into the simulation. However, realistic depictions of physiological deformations are challenging, due to the varying thickness and material properties of the vessel wall and external tissue support [6].

Bäumler et al. [6] improved simulation results in comparison to 4D flow MRI, by using deformable models. More realistic deformations could be obtained, compared to previous non-rigid simulations.

1.3 Flow Visualization

Flow visualization has been a research topic for hundreds of years. Still, there are challenges in the visualization of flow data, particularly in 3D, that could not be overcome yet. Fluid simulations and the subsequent data visualisation is becoming more important in many fields of engineering and medical applications. Also, development of computer hardware provides the computational power to generate ever more detailed and dense datasets. However, effective visualization of dense vector fields is still challenging. Visualization of raw 2D data can be done effectively, as shown by Urness et al.[31], using a texture-based visualization in conjunction with multiple color maps to convey directional data as well as different flow properties. Useful depiction of raw 3D data, on the other hand, is more difficult. As dense 3D vector fields cannot be fully displayed on a 2D plane, i.e. a monitor, different techniques have to be used, to visualize important features of the dataset in a useful manner.

1.4 Goal of this Thesis

The goal of this thesis is to explore different flow visualization techniques with respect to aortic dissections. There are many techniques used in different fields to produce effective visualizations. Their applicability to this particular use case, however, needs to be investigated, as analysis of hemodynamic flow poses several difficulties. We combine several methods to aid the analysis of flow in aortic dissections, and develop new visualizations tailored to this application.

2. State-of-the-Art

The analysis of fluid dynamics, flow patterns and the associated physical values is widely adopted in certain areas of engineering. Modern aeroplane, automotive or fluid transportation design depend largely on simulation and visualization of fluid dynamics. In recent years, the importance of flow analysis of the human vascular system has gained importance, as modern instruments allow for detailed insight into flow characteristics, without the need for an intervention. Despite that patient-specific flow data is not routinely acquired [6], not even during treatment of life-threatening cardiovascular deceases. Data collected during a 4D flow MRI sequence measures blood velocity over a specific time span, typically a cardiac cycle. Commonly, raw 3D data is visualized with maximum intensity projection (MIP), color-coding velocity. As shown in Figure 2.1, this is useful to identify bulk flow and low volume flow areas, also showing retrograde flow during diastole quite effectively [1]. Another common way is to compute streamlines or pathlines, which is not a direct visualization of the raw data, but rather one of many techniques to better visualize flow.

2.1 Flow Visualization

Visualization of complex flow datasets is not trivial and requires additional processing to produce effective visualizations. Different approaches to visualize flow have emerged, in an effort to overcome the difficulties associated with visualizing dense 3D flow data. In this section a basic introduction into different flow visualization techniques and the challenges, that occur is given.

As mentioned before, the simplest representation of a vector field is **direct visualization**. Without editing the original data or generating new structures, a vector field is visualized with volume rendering, as shown in Figure 2.1. These visualizations often result in cluttered and occluded representations of the data, because they display all the available data simultaneously. Another group of visualization techniques are **texture-based**. A noise texture is created and deformed/morphed using the local vector field. This technique yields a dense representation of the flow in a specific area but cannot be easily extended to 3D.

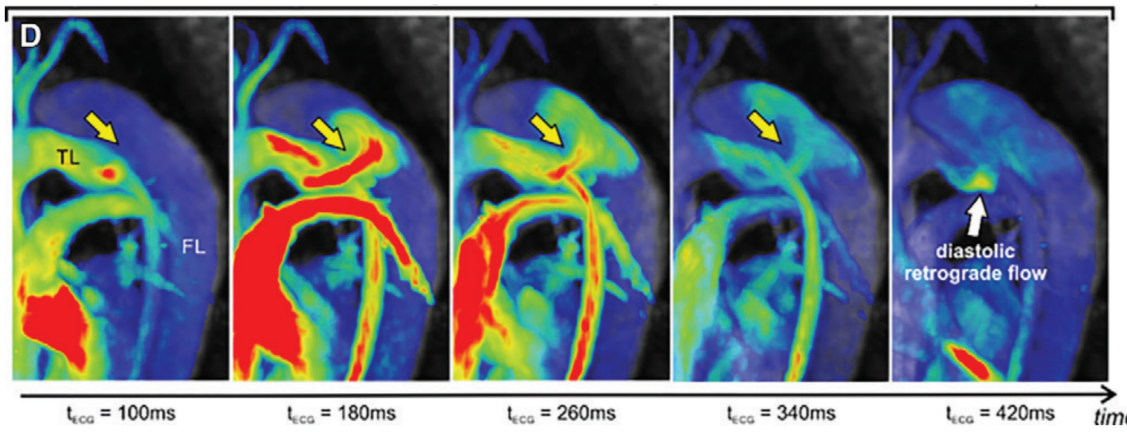


Figure 2.1: Velocity color-coded MIP of a 4D flow MRI data showing a Type B aortic dissection at the late systolic phase [1].

Geometric visualizations are usually curves, surfaces or volumes generated from a vector field. These geometric structures can reduce clutter in visualizations significantly if used correctly and are commonly obtained by integration, simulating particle movement in flow fields. This technique is covered in Section 2.2.

To further improve the effectiveness of a visualization, **feature-based visualizations** can be used to segment the dataset, discarding irrelevant data and only display areas, that may be important to analyze. These areas can be defined by location or by flow characteristics. Köhler et al. [32] proposed a robust method of extracting vortices in 4D PC-MRI flow data using pressure information. In another work, they present a visualization for eccentric flow jets in great arteries [33].

2.2 Integral Structures

Streamlines are one of the most common and most intuitive techniques to visualize flow. They are part of a group called integral structures, that can be extracted from flow data. Integral structures are based on simulated particle movement through a vector field, which can be described as follows [34]. Given a dense set of massless particles with indices i moving in the spatial domain $\Omega \subseteq\rightarrow \mathbb{R}^n$, an n-dimensional *steady flow* v is typically described as a differential equation of particle locations $x_i \in \Omega$ with respect to time $t \in \Omega$:

$$\frac{d_i(t)}{dt} = v(x_i(t)). \tag{2.1}$$

Steady flow is indicated by a vector field of instantaneous velocities, that is constant over time. On the other hand, *unsteady* flow changes over time. Particle motion for *unsteady flow* can be described by:

$$\frac{d_i(t)}{dt} = v(x_i(t), t). \tag{2.2}$$

Figure 2.2: Leonardo da Vincis drawings of vortices generated at the aortic valve [2].

Streamlines are an integral structure and can be obtained by solving Equation 2.1 using integration. *Pathlines* are particle traces in *unsteady flow*, generated by solving Equation 2.2. Another integral structure, the *streakline*, is obtained by continuously seeding particles at the same location in *unsteady flow* and combining them into one line. *Timelines* are similar to *streaklines*, except they are not seeded at the same location, but rather on a line or curve. All of the mentioned integral structures are lines generated from flow data and are, therefore, called integral lines. They can also be extended into higher dimensions (integral surfaces, volumes).

2.3 Illustrative Visualization

Analysis and visualization of flow is not a new concept. Rather it has been a research topic for at least 500 years. Leonardo da Vinci was the first to analyze flow patterns in water and while investigating the human cardiovascular system and performing hemodynamic studies of heart and valve motion [2]. Around the year 1500 da Vinci was already able to recognize the existence of aortic valve vortices and visualize them in the drawings shown in Figure 2.2. Da Vincis drawings are the first recorded visualizations of flow and were drawn without MRI measurements, fluid simulations or any other electronic assistance.

Contemporary research is working on computer generated illustrative visualizations, as they have several advantageous properties. These properties help the effectiveness of a visualization and become apparent when comparing illustrative visualizations to traditional visualization. Illustrations depict important flow features without showing all of the fluid movement around them. This reduces clutter and obstruction and helps the viewer to focus on relevant information. Illustrative visualizations also often use shading techniques, that improve the perception of shape, depth and curvature. They can also include arrows or writing to give more information about a specific flow feature. The goal of Illustrative visualizations is to generate intuitive representations of flow, that can convey useful and relevant information effectively, rather than producing photo realistic renderings.

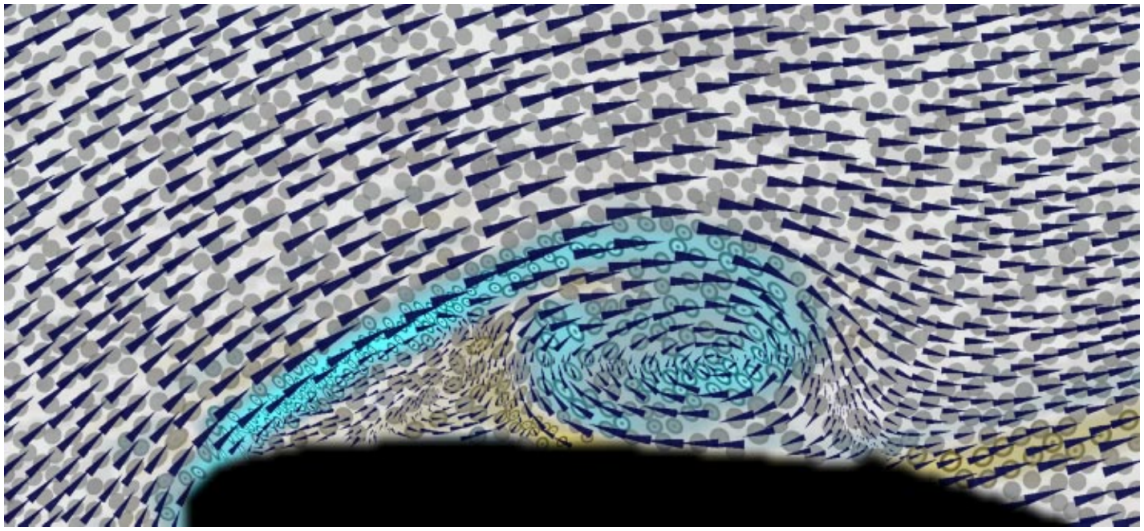


Figure 2.3: Visualization of dense flow data around an aerofoil. Velocity represented by arrow glyphs, vorticity by color and strain by ellipse glyphs [3] ©1999 IEEE.

Perceptual effectiveness of a visualization refers to a visualizations ability to display information in a way, that can be understood by the viewer [34]. Improving perceptual effectiveness contributes to a visualization that can better display multiple measures (velocity, pressure, etc.) simultaneously, while not overwhelming the viewer. Common techniques to improve visualization of raw data are the use of glyphs and color-coding. You can see both techniques applied in Figure 2.3 enabling simultaneous visualization of velocity using arrow glyphs, vorticity using color and strain using ellipse glyphs. The arrows used differ from traditional arrow plotting, in that they also show local flow velocity with arrow size and shape. The ellipse glyphs behave similarly and reflect rate of strain, divergence and shear in ellipse radii, area and eccentricity respectively. The perceptual effectiveness of 3D visualizations can also be enhanced using similar techniques. Analogous to 2D representations, color is commonly used to show velocity, as shown in Figure 2.1.

Another important aspect of perceptual effectiveness is occlusion and cluttering. Visualizations with overwhelming amounts of integral lines or glyphs can degrade its effectiveness. When generating integral lines, choosing an appropriate amount and also suitable seed locations is critical to the success of a visualization. Displaying too many lines can lead to important flow features being occluded, while displaying too little or the wrong lines may cause important features to not be visible at all. A method for choosing suitable seed points, that produce good coverage of an area of interest was proposed by Behrendt et al. [35].

2.4 Challenges

During the development of flow visualization techniques, a lot of effective solutions for the presented challenges could be found. But as fluid simulations and the visualization thereof are becoming more complex, new challenges emerge.

Comprehensive CFD simulations can generate large amounts of data. While these calculations take a lot of time, visualizations benefits from being done as quick as possible, ideally in real time. Accessing massive amounts of data to generate an image can induce significant latency and harm the overall experience. Also, fluid simulations often output their results in unstructured grids. This slows down the visualization process even more, as the position of vectors in a vector field is required for many flow visualization techniques. To work with unstructured grid, we use an additional data structure for efficient access. Visualizing flow in 3D rather than 2D not only adds computational complexity, but requires visualization techniques to cope with occlusion, occurring when integral structures overlap, and cluttering, because dense 3D data or structures are difficult to display effectively. Representations of unsteady flow add another layer of complexity, as there is another dimension of data, which needs to be conveyed effectively. Integral structures can be color-coded to simultaneously display spatial and temporal location, but the most common way to visualize time-dependent or movement data is animation. Along with animating time-dependent data using moving lines, glyphs or particles, animation can also be used to visualize complex static data, by moving the camera, clipping planes or changing opacity over time [36]. Pathlines and streaklines can also give information about flow characteristics, that change over time without the need of animation.

3. Methodology

The medical visualization framework Visician, developed by Dr. Gabriel Mistelbauer, serves as basis for the implementation of the following visualization techniques. The provided user interface features and session management, as well as algorithms for skeletonization and extraction of the vessel tree are used for flow visualizations. Loading, saving and manipulation of simulation data is partly handled by the visualization toolkit (VTK).

3.1 Data Acquisition and Format

Flow visualizations can be performed on different kinds of data. The data used for the visualizations shown in this thesis were obtained using CFD simulations [6]. A model of the aorta together with hemodynamic measurements of patients suffering from aortic dissection were acquired using computer tomography angiography (CTA) and 4D flow MRI. Captured velocities and estimated flow rates at different locations in the aorta and branching arteries were used as boundary conditions for the simulation. Following simulations using deformable wall models were performed for 4000 timesteps over one cardiac cycle. Every timestep is saved in a .vtu file containing the entire grid structure as well as velocity, pressure, wall shear stress and several other values per point.

3.2 Preprocessing

A single timestep is sufficient to perform any of the visualization techniques in this thesis. As mentioned before each timestep contains the entire grid, including every points position and a cell array containing indices that refer to the points, that make up each cell. Cells are sets of three points, that each form a triangle, while all points are interconnected by cells to form a grid of tetrahedrons. The point positions need to be specified in the dataset, because the points are not aligned on a regular grid, but rather distributed arbitrarily. The dataset classifies each cell by domain, being either fluid (blood), flap (dissection flap) or outer (vessel wall).

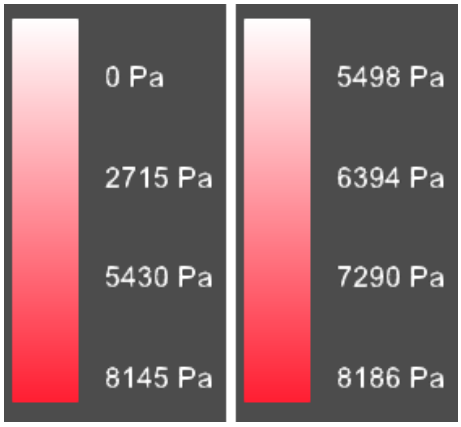


Figure 3.1: The color scale legend for pressure values in pascal with windowing functions: center = 0, width = 100 and center = 34, width = 33.

This is used during simulation to differentiate blood from vessel material, but also to assign different properties to regular vessel wall (outer domain) and dissection flap (flap domain). We can also use these domains to generate different meshes and visualize the vessel wall, dissection flap and fluid differently.

Displaying the fluid, flap and outer domain surface meshes is not trivial, because the data does not indicate whether a cell is on the surface of an object or on the inside. Rendering the cells with shading generates various artifacts, because point normals cannot be correctly calculated without selecting only surface cells first. To produce an accurate representation the surface faces of each domain need to be extracted, requiring less faces, that need to be rendered and allowing for calculation of point normals. Both surface extraction and normal calculation are performed using VTK.

3.3 Color-Mapping

Along with flow characteristics, there are several different kinds of information, acquired during simulation, that are critical to analyzing hemodynamics. In addition to velocity, values for pressure, wall shear stress, traction and displacement were recorded for every point in the model. Although pressure is given for every vertex, the vessel wall and the dissection flap are areas where the analysis of this value is most important. Therefore, a color mapping on the surface mesh is used for this visualization.

In conjunction with the visualization, a legend, shown in Figure 3.1, including a color scale and the according minimum and maximum values, as well as intermediate values, is displayed to allow the viewer to relate color to physical values. Because the pressure differences along the aorta and between lumina can be significant, a detailed analysis of a specific area is difficult while coloring over the entire pressure range. To mitigate this issue, the colors can be adjusted using a windowing function. At startup, the windowing function is set to zero for the center and 100 for the width, meaning no offset and 100% of the pressure range being displayed. By adjusting the center and width, the desired pressure range can be selected and analyzed in detail. Adjusting the windowing function updates both the color mapping and the legend.

3.4 Flow Visualization

One of the simplest ways of direct visualization of velocity vectors is using simple line glyphs. To produce this visualization a set of lines is drawn, originating at every vector's position and extending along the vector's direction for a specific length. The line length can easily be adapted to the vector's length too, displaying velocity in the flow on a very rudimentary level. To increase perceptual effectiveness lines are also colored relative to their length. To reduce occlusion and cluttering the amount of displayed lines can be adjusted. Because of the persisting issues with direct visualization of flow data, a different technique was employed additionally to better show flow patterns in the datasets.

Integral structures have proven to be effective at representing flow. Given by the available data and the chosen visualization goal, integral line structures were selected. Pathlines are often used to show blood flow over time. Incorporating multiple time steps of the simulation, the generation of pathlines is possible. However, determined by the fact, that we only load one timestep at a time, the only integral line structures, that can be produced, are streamlines.

Traditionally the acquisition of a streamline starts by selecting a suitable start location, called seed point. Seed points need to be located inside parts of the vector field where flow occurs, i.e. not inside volumes, where all velocity vectors are zero. In addition to this requirement, there are different aspects to selecting seed points, as they affect the location of the streamlines and the way they distribute.

3.5 Seed Point Selection

In case of flow visualizations involving the aorta, lines are most frequently seeded at the aortic root. Analyzing flow at one specific timestep this location is not necessarily the best. As we most often need to analyze the flow at certain locations along the aorta, generating streamlines locally, by growing them starting at a specified location, is desirable.

Another criterion for seed point selection is line distribution. Lines seeded in the center of the lumen may distribute differently, compared to lines seeded at the vessel wall. To avoid missing possibly important flow patterns, lines should be seeded evenly over the entire cross-section of a lumen.

In order to create seed points that meet the previously stated criteria, the cross section of the vessel is required. The cross section can also be described as the intersection of the vessel volume and a plane, that is perpendicular to the local course of the vessel at the point of intersection. The course of a vessel can be estimated using a centerline, which represents the primary flow path of a vessel.

3.5.1 Centerline Extraction

To analyze the flow inside the aorta, the seed plane needs to be a cross-sectional plane of the aorta. Therefore, a centerline of the main vessel, excluding any branching vessels or artifacts needs to be extracted. To acquire a centerline the vessel is skeletonized by voxelizing the vessel volume and subsequent erosion and then the vessel tree extracted. These algorithms were already implemented in the framework and are briefly explained below.

First, the volume needs to be voxelized. The purpose of voxelization is to transfer the irregular volume mesh given in the data into a regular grid of voxels, that can be skeletonized. In this process first a grid of certain voxel size is created and aligned whith the volume mesh of the vessel. Afterwards the voxels containing a part of the mesh are set to value 1, the rest to 0. This results in a 3D binary dataset of the vessel volume. The next process used is called erosion and aims to thin the volume until only the centerline of any vessel or lumen remains. During erosion specific voxels of value 1 are iteratively turned to value 0, reducing the volume without changing its topology. Thinning is performed until only lines of single voxels remains. Finally, the lines of voxels are assembled into line segments. Each line segments spans from its beginning to a location where multiple lines meet in a bifurcation or between bifurcations. The result is a set of unordered lines composed of points with regular distance between them.

Building upon the processes already available in the framework, the vessel tree generated needs to be processed further to obtain a usable centerline of the aorta. As the lines in the vesseltree are not ordered and also include a lot of branch vessels and some artifacts, the line segments associated with the aorta need to be selected and assembled into a joined centerline. In the tests performed the longest line segment was found to be a section of the descending thoracic aorta. Starting with this segment, adjacent segments are selected, if they meet two criteria. The beginning of a continuing segment must not exceed a prespecified distance from the current end of the centerline and must not exceed an angle of 60°. If multiple segments meet these criteria, the one with the smallest angular discrepancy is selected. If no continuing line segment can be found, the centerline ends.

Using skeletonization, followed by our selection process, a centerline for the aorta can be obtained. While testing on three different datasets, the algorithm always yielded a centerline at the descending aorta, for both lumina. Coverage of the vessel structure was mixed, with one centerline extending into the leg arteries, while the remaining two results did not extend as far. Overall, centerlines extended along most of the descending aorta. The generated lines where mostly continuous, but gaps did occur. These gaps are caused by artifacts of the skeletonization, that produce segments connecting both lumina. Segments falsely connecting lumina are skipped during centerline assembly resulting in small gaps in the centerline. This is not such a big problem for the selection of seed points, as it is for volume trimming. A gap in the centerline can cause the volume selected for streamline generation to be separated, not allowing lines to pass through the gap. A possible solution is filling the gaps, by applying a b-spline to the selected centerline segments. In the test, however, the discontinuities turned out to be small enough to not cause problems during trimming.

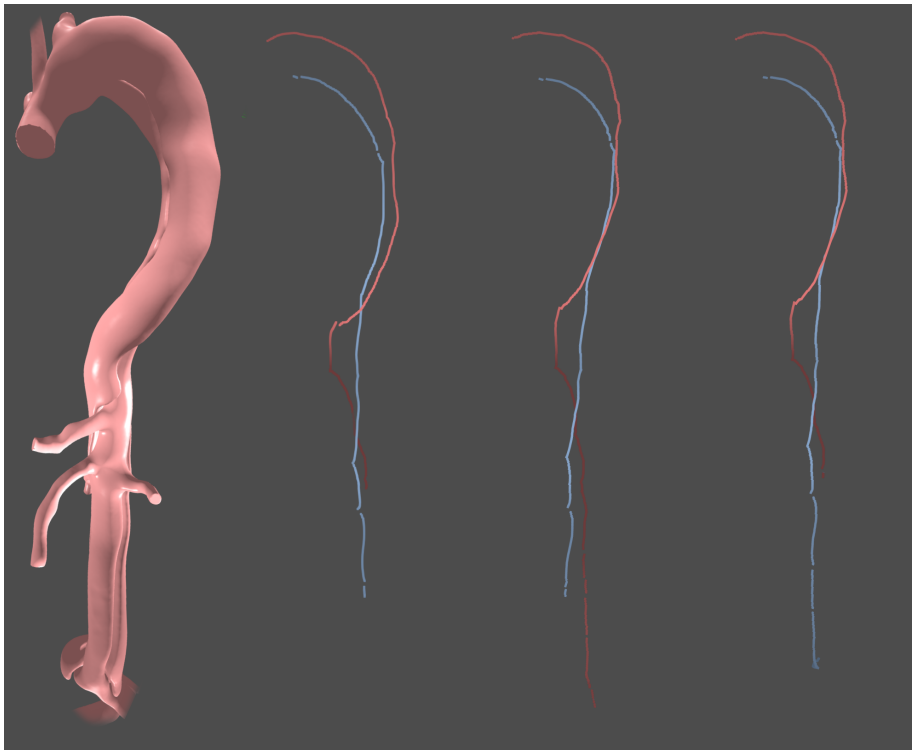


Figure 3.2: Result of centerline extraction for three different datasets, compared to surface mesh of the aorta.

Because of the small distance between true and false lumina, the skeletonization result contained many paths in between lumina. These are not real flow paths however and need to be avoided during assembly of centerlines. In the tests, the centerlines generated individually for each lumen were always separated and specific to the aorta, neither deviating into a branch vessel nor into the other lumen. The centerlines generated for different datasets are shown in Figure 3.2. Points on the centerline are rendered individually, showing the course of the line as well as possible gaps, which would not be clearly shown when rendering the centerline using a continuous line.

3.5.2 Seed Area

Acquiring the centerline of the vessel, provides the necessary data we need to generate a cross-section. The centerline generated using the methods described in the previous section is given as a list of point locations from start to end of the centerline. Any of those points can be chosen as location of the seed area, which is generated as follows. First, the plane perpendicular to the centerline is needed to place seeding locations. Two vectors, that describe the perpendicular plane can easily be found using the dot product. The vector $\vec{v}_c$ along the centerline, locally to the selected seeding location, is calculated by subtracting the selected points location by the next ones. Next, perpendicular vectors $\vec{v}_1$ and $\vec{v}_2$ are calculated using the dot product : $\vec{v}_1 = \vec{v}_c \cdot \vec{v}_a, \vec{v}_2 = \vec{v}_c \cdot \vec{v}_1$ ($\vec{v}_a$ being an arbitrary vector). Vectors $\vec{v}_1$ and $\vec{v}_2$ describe the plane of the local vessel cross section and are used to generate points on this plane.

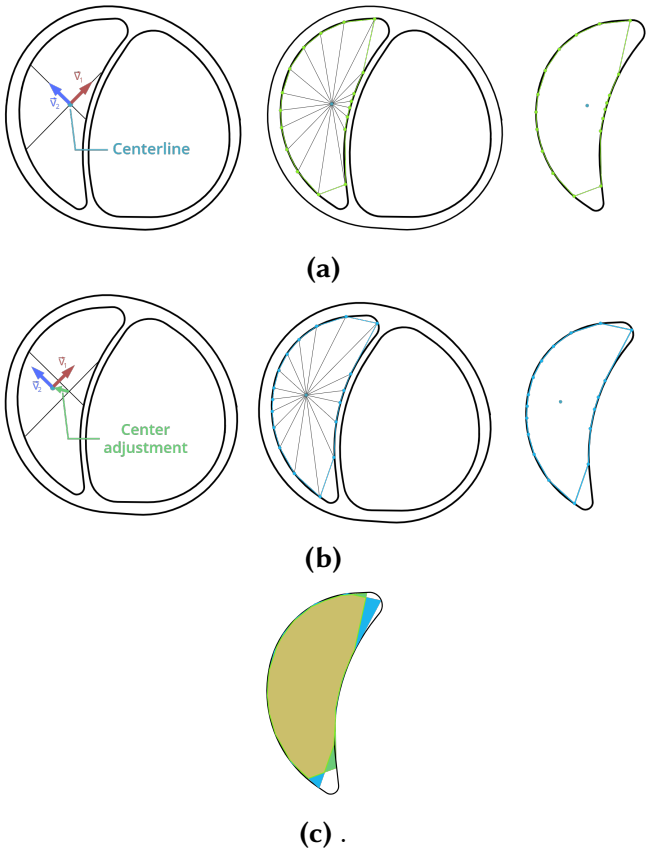


Figure 3.3: Concept of ray casting method for finding the border of the vessel. **(a)** An example of the resulting border-polygon of a concave lumen cross section with an uneven distribution of polygon vertices on the concave profile. Also regions in the narrow part of the cross section are not included at all. **(b)** By shifting the ray casting origin, the vertices of the border polygon are distributed more evenly. This produces a more accurate estimation of the actual lumen border. **(c)** Comparing the polygons produced with and without center adjustment, you can see, that adjusting the ray origin results in better fill of narrow areas while also reducing polygon area outside the actual lumen.

Streamlines can not be seeded outside of volumes, where flow occurs. Specifically, this means seed points can only be located inside the fluid domain. After calculating the cross-sectional plane, the border of the fluid domain in this cross-section needs to be found. In order to find the border, an algorithm measuring radial distance between centerline and vessel wall is used. The center of the cross-section is given by the selected seeding position on the centerline. First rays are cast in positive and negative $\vec{v}_1$ and $\vec{v}_2$ direction respectively. This allows measuring the approximate mean radius of the vessel, which is needed to calibrate parameters used in the next step. Also by measuring the rays lengths and comparing the ones along the same axis, a center adjustment can be performed. Due to the nature of the process used to generate the centerline, it may not be perfectly in the center of the vessel. The irregular and sometimes concave shape of the cross-section of a lumen reduces the accuracy of our method. By adjusting the center, a better estimation of the border can be achieved as shown in Figure 3.3c.

The next step towards generating a set of seed points is building an approximation of the lumen border, that acts as bounds while generating seed points. Similar to the previous step, rays are cast from the adjusted center outwards. The distance at which each ray intersects the vessel wall is saved and limits the radius seed points can be placed on for that specific direction from the center. The angle between rays is determined by the mean radius measured during the center adjustment step. With greater radius, the angle between rays has to increase to maintain a roughly constant sampling frequency on the vessel wall.

Intersection between ray and vessel wall is handled by checking proximity of a ray and cells of the mesh. Starting at the direction of vector $\vec{v}_1$, the ray is advanced away from the center. Every step, the distance between the tip of the ray and the closest face of the vessel wall mesh is checked. If the distance falls below the distance between points in the cell, the ray ends. This process is repeated several times, adjusting the angle of the current ray until a full revolution is complete. Finding the closest cell in the mesh involves checking every single cell if the data is not arranged or sorted. To drastically increase performance, cells of the vessel mesh are organized in a grid. The grid structure defines isotropic chunks (cubes with uniform edge length) of specific size in a grid aligned with the data and sorts cells into the local chunk. Because the chunks are arranged in a regular grid, selecting the chunk local to any 3D-location inside the grid can be done in constant time. This reduces the amount of cells, that need to be checked, dramatically. Because the tip of the ray can be at the edge of its local chunk however, the neighboring chunks have to be checked as well.

With the seed plane found and the bounds defined, all requirements for generating a set of seed points are met. This algorithm is similar to the border-finding algorithm, in that vectors rotated by a specific angle every step, are used to build the grid of points. However this time the radius is not advanced (length of the ray) before advancing the angle. Instead all points on one radius are placed before advancing to the next one, effectively building shells of points around the center point.

Before generating the seed points, point spacing and border margin need to be specified. The point spacing s determines the distance between points and consequently the amount of seed points, while the border margin m determines the minimum distance between any point and the border. As mentioned before, points are generated by placing concentric rings around the adjusted center and seed points on those rings. Important for this process is point spacing and border proximity detection.

To generate a grid of roughly equidistant points, they are placed on rings of radius r with $\{r = n \cdot s \mid n \in \mathbb{N}, n \geq 0\}$, n being the ring index. To ensure equal distance of points on a ring, the circumference C is divided by s resulting in the amount of points p placed on the ring. This value is rounded down, because the distance of points on a circle is less than the length of the circular segment between them. With the amount of points on the circle, 360° is divided by p, determining the angle α between points, which is needed to calculate the position of each seed point.

$$\begin{aligned} x &= a + r\ \cos\alpha, \\ y &= b + r\ \sin\alpha. \end{aligned} \tag{3.1}$$

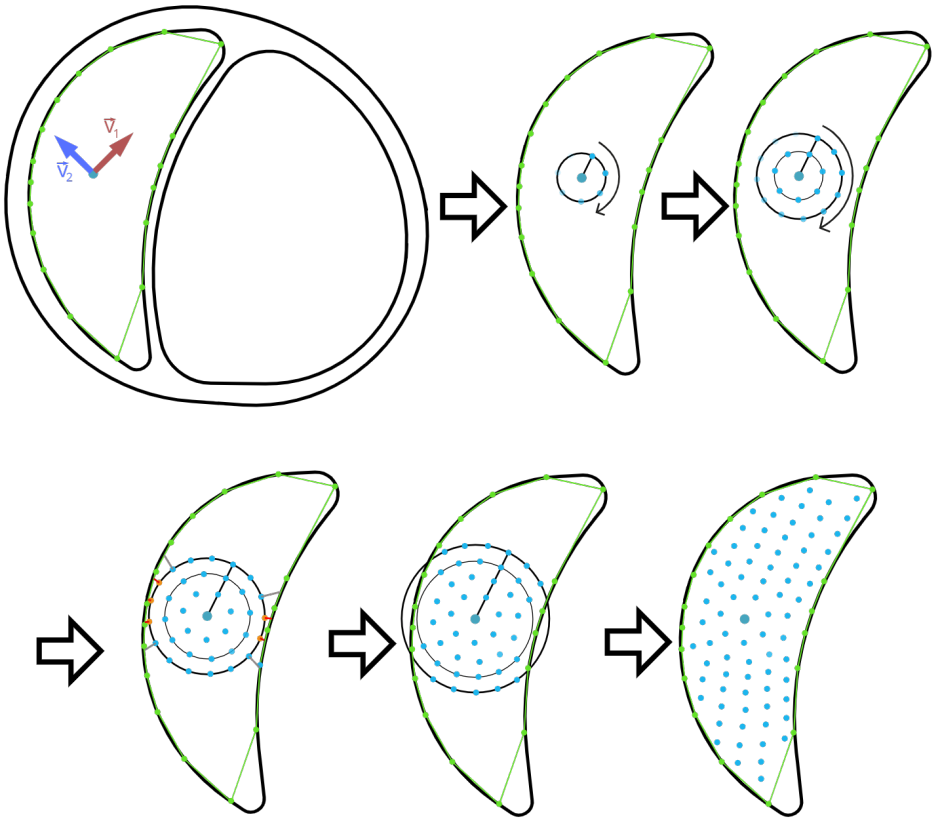


Figure 3.4: Illustration of the Seed point distribution process. Points are placed on concentric radii with the same distance between radii and points on the circular path. Points, which would appear too close or beyond the border are discarded.

The parametric form of the circle equation in a Cartesian coordinate system, you can see in Equation 3.1, is used as basis of our equation.

$$\begin{aligned} \vec{v}_x{}' &= \vec{v}_c + \vec{v}_x \cdot r \cos\alpha, \\ \vec{v}_y{}' &= \vec{v}_c + \vec{v}_y \cdot r \sin\alpha. \end{aligned} \tag{3.2}$$

Equation 3.1 can be altered, so it can be used in 3D space. However for circles, that are not on a plane parallel to the XY-plane, a set of two perpendicular vectors is needed. Equation 3.2 are the equations adapted to use vectors, but neither $\vec{v}_x{}'$ nor $\vec{v}_y{}'$ produce the correct location of a point on a circle with center at $\vec{v}_c$ and radius r.

$$\vec{v}_p = \vec{v}_c + (\vec{v}_x \cdot r \cos\alpha) + (\vec{v}_y \cdot r \sin\alpha). \tag{3.3}$$

In order to calculate the correct position, the equations for $\vec{v}_x{}'$ and $\vec{v}_y{}'$ need to be combined into Equation 3.3. With perpendicular, normalized vectors $\vec{v}_x$ and $\vec{v}_y$, the circle equation can be used, because the circle is placed on a 2D plane defined by the two vectors, with $\vec{v}_x$ equivalent to the x-axis and $\vec{v}_y$ equivalent to the y-axis.

Using this approach, the location $\vec{v}_p$ of a point on a circle with radius r can be calculated by adding the x and y shift on the circle plane to the center location $\vec{v}_c$.

$$\vec{v}_p = \vec{v}_c + \vec{v}_1 \cdot r \cdot \sin(\alpha \cdot m) + \vec{v}_2 \cdot r \cdot \cos(\alpha \cdot m)\ , \quad 0 \leq m < p. \tag{3.4}$$

Finally Equation 3.4 is used to calculate locations on a circle around the adjusted center. Index m is used to advance the angle and distribute points evenly on the circle. Increasing m for every point until one revolution is complete and subsequently increasing the radius using the method described above, generates points from the center expanding outwards with roughly equal distances. You can also see an illustration of the seed point generation process in Figure 3.4.

As previously mentioned, not all points on every ring are suitable seeding locations. For this reason, points outside the border are discarded. Essential for this step are the individual ray lengths, acquired during border finding. The rays are set up with equal angles between them. This allows us to simply save the length in a sorted list and calculate the angle between the ray and $\vec{v}_1$ based on the angular difference β between rays and their indices.

When generating points, the angle between the vector from $\vec{v}_c$ to $\vec{v}_p$ and $\vec{v}_1$ is given by $\alpha \cdot m$. Dividing the result by β, rounding up and down, returns the indices i_1, i_2 of the two rays, with lengths l_1 and l_2, either side of the point.

$$\begin{aligned} \text{ratio} &= \frac{(m \cdot \alpha) - (\beta \cdot i_1)}{\beta}, \\ r_{\max} &= \text{ratio} \cdot l_1 + (1 - \text{ratio}) \cdot l_2. \end{aligned} \tag{3.5}$$

By interpolating between those two rays using Equation 3.5, a maximum radius can be found for any possible angle between $\vec{v}_1$ and a point. A visual representation of the border proximity detection can be seen in Figure 3.5

Adjusting parameters for seed point spacing and border margin result in different amounts of points as well as different arrangements. Various examples for sets of seed points generated at the same location, but with varying spacing and margin are shown in Figure 3.6.

3.6 Streamline Generation

With the seed point selection complete, the next step in visualizing flow using streamlines is generating the line structures. This is done by iterating over the vector field produced during simulation. However, because the simulations use deformable models, meaning the mesh of the solid domains outer and flap are not static and move as well, the vectors inside the vessel wall and flap are not zero. Furthermore, as the dissection flap is moving laterally to the primary flow direction, the blood in direct proximity of the flap follows this movement resulting in velocity vectors close to perpendicular to the vessel wall or dissection flap. After testing, this was found to be the cause of streamlines advancing into volumes of solid domains, effectively penetrating the dissection flap. This behaviour suggests cross flow between true and false lumen, but can occur at locations without holes in the dissection flap (fenestrations), producing an incorrect visualization.

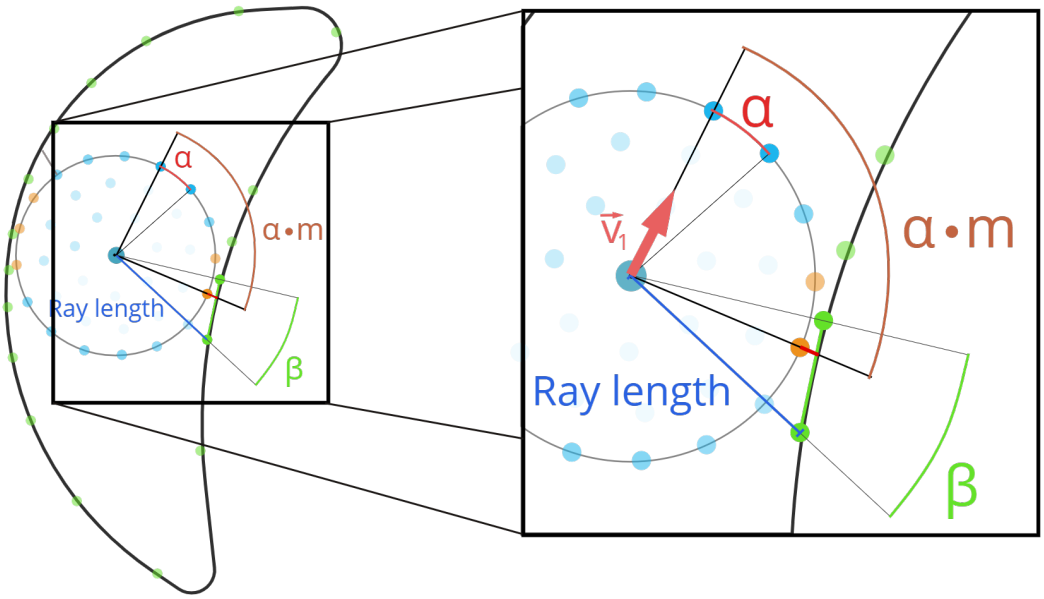


Figure 3.5: Illustration of our proximity detection algorithm. In this image accepted seed points are colored blue, while discarded points are marked orange, as they are too close to the border (green). This distance is calculated from the current radius and the interpolated ray length between center and border.

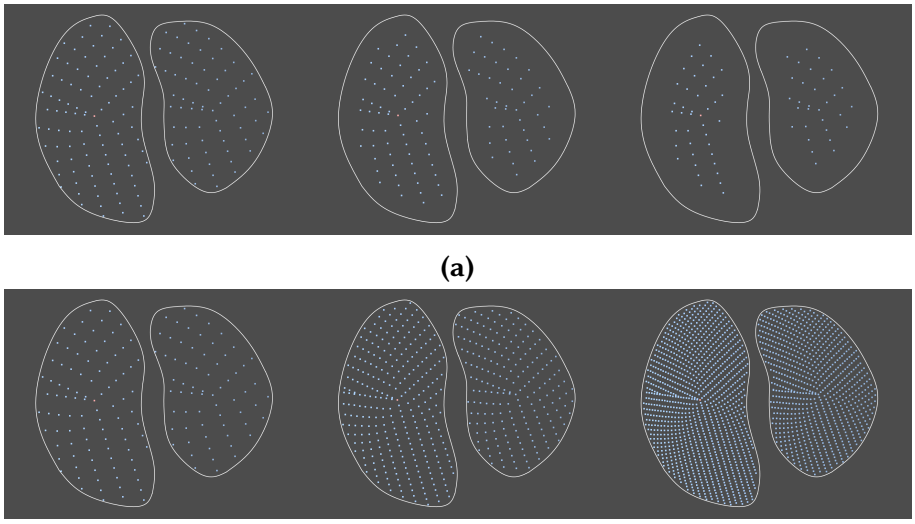
3.6.1 Vector Field Trimming

To mitigate this problem, the field used to generate streamlines is trimmed to only include vectors inside the fluid domain. This prevents the line building algorithm from using vectors located inside a solid domain, but does not fix the flow perpendicular to the vessel wall inside the fluid domain.

For the seed point selection process, surface mesh cells needed to be organized to increase performance, because the simulation data is presented in an unstructured grid. The same problem of execution time arises when generating streamlines. Both algorithms used to compute the line structures involve finding the closest vector to a given location. Therefore the vector field is organised in the same volumetric grid structure as above. Only instead of mesh cells, individual points including the corresponding velocity vector are sorted into isometric chunks in a regular grid. While filling the grid, points outside the fluid domain are discarded, trimming the vector field to only include fluid flow vectors and exclude any solid deformation vectors.

Although trimming the vector field to only include the fluid domain reduces the frequency of streamlines ”jumping” between lumen, it does not completely eliminate the issue. To isolate the lumina and restrict line structures to only be generated in either one of the lumina, a lumen specific trimming algorithm is used.

Goal of the algorithm is to create two separate volume grids, one for each lumen and perform the streamline generation separately on each dataset. This opens up the option of displaying two sets of streamlines to visualize the flow of both lumina simultaneously and possibly show interaction of both flows at entry and exit tears.



(a)

(b)

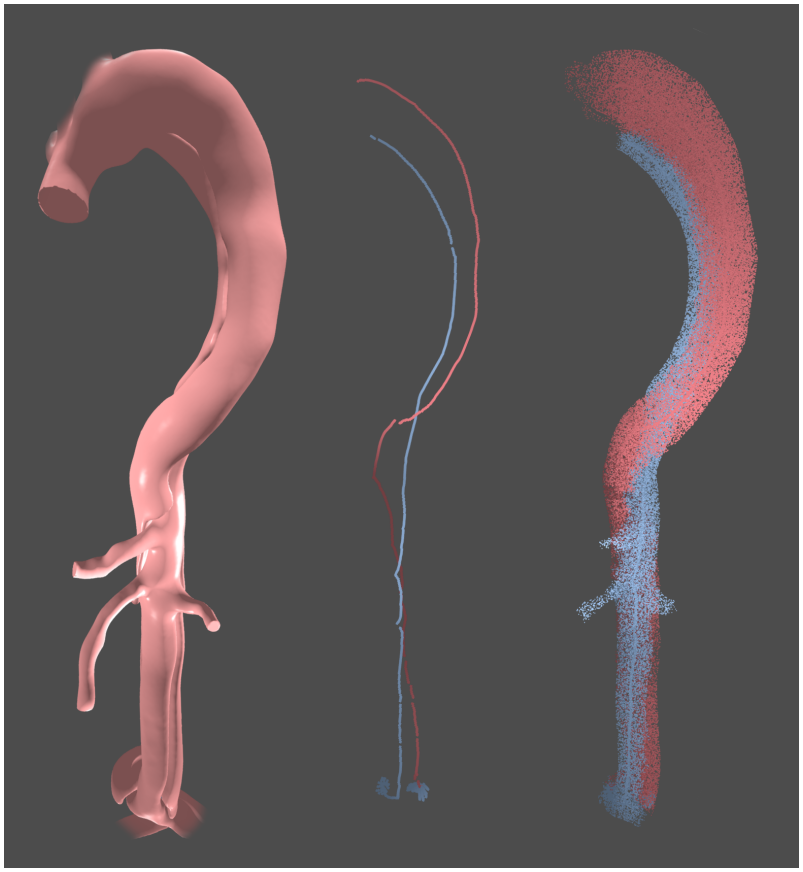
Figure 3.6: Results of the seed point algorithm demonstrating the effect of margin and spacing parameters. **(a)** Seed points generated with increasing border margin left to right. **(b)** Seed points with decreasing spacing left to right. (White outline added for better comparison)

The first step to create the two datasets is to copy the pre-trimmed volume grid. In order to isolate the vectors relevant for a lumen, chunks of the volume grid containing values of this lumen are collected. Afterwards all chunks outside of the lumen are discarded. The seed area approach described in Section 3.5.2 is used to select the chunks relevant to a lumen. By building the border for every point on the centerline of a lumen and collecting the IDs of all chunks, that were entered during the process, all the chunks, that contain vectors relevant to the flow of this lumen can be obtained. Subsequently all chunks of the volume grid, except for the collected ones, are cleared.

To verify accurate trimming of the vector field and complete selection of relevant chunks of the grid, all points selected to be used during streamline generation in a lumen were rendered and inspected. The result closely resembles the shape of the lumen and can be compared to the fluid domain mesh. A visualization of the trimmed field can be seen in Figure 3.7.

3.6.2 Line Structure Computation

Streamlines are integral structures, acquired using integration. Two different calculation methods were implemented for the line building algorithm, both iteratively extending streamlines based on the direction and magnitude of local velocity vectors. The process of building a streamline starts at the seed point. From this location, the nearest velocity vector is searched using the grid data structure. Based on the data in the vector field, the position of the next point of the streamline is calculated. The distance between points can be chosen to be constant or relative to the magnitude of the vectors used for calculation.



(a) **(b)** **(c)**

Figure 3.7: Results of lumen specific trimming. **(a)** fluid domain mesh. **(b)** centerlines. **(c)** selected points for true and false lumen in red and blue.

For this work, the distances were chosen to be calculated based on the vector length, because this enables easier visualization of velocity in a later step. Calculation of the next point of the streamline is done until a maximum number of steps, specified beforehand, is reached.

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \Delta t \cdot \vec{v}(\vec{x}(t)) \cdot \text{step}. \tag{3.6}$$

The first method of calculating the next position, the **Euler method**, only uses the nearest velocity vector $\vec{v}(\vec{x}(t))$ and the current last position $\vec{x}(t)$ to calculate the next point position $\vec{x}(t + \Delta t)$, as shown in Equation 3.6. The constant step is used as a factor for adjusting step length (multiplier for distance between points).

This is a simple and fast way to produce streamlines, but lacks accuracy. Only using one vector as input, the representation of flow diverges from the actual flow pattern, especially in circular flows like vortices. This effect is visualized in Figure 3.8c.

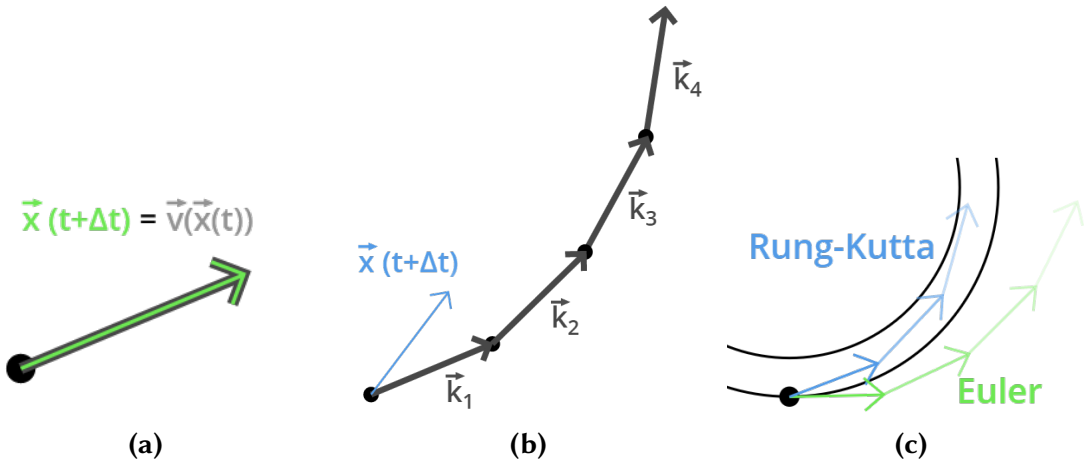


Figure 3.8: Illustration of two streamline computation methods of different accuracy and computational expense. **(a)** Visualization of the Euler method with $\vec{v}(\vec{x}(t))$ in grey and the resulting $\vec{x}(t+\Delta t)$ in green. **(b)** Illustration of 4th-order-Runge-Kutta integration with the vectors $\vec{k}_1 - \vec{k}_4$ of the vector field in grey and the calculated $\vec{x}(t+\Delta t)$ in blue. **(c)** Comparison of the streamlines generated by both methods in case of circular flow. The Runge-Kutta method (in blue) manages to more closely match the actual flow, while the Euler method produces a significantly diverging streamline.

$$\begin{aligned} \vec{x}(t+\Delta t) &= \vec{x}(t) + \frac{1}{6}\cdot(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \quad \text{with} \\ \vec{k}_1 &= \Delta t \cdot \vec{v}(\vec{x}(t)), \qquad \vec{k}_2 = \Delta t \cdot \vec{v}(\vec{x}(t) + \frac{\vec{k}_1}{2}), \\ \vec{k}_3 &= \Delta t \cdot \vec{v}(\vec{x}(t) + \frac{\vec{k}_2}{2}), \qquad \vec{k}_4 = \Delta t \cdot \vec{v}(\vec{x}(t) + \vec{k}_3). \end{aligned} \tag{3.7}$$

The more accurate method is **4th order Runge-Kutta integration**, albeit more computationally elaborate. With this method four vectors are used to determine the position of the next point of a streamline. First the nearest vector $\vec{k}_1$ is used to determine a position. The vector $\vec{k}_2$ closest to this position is then again used to determine the next one, building a chain of four vectors : $\vec{k}_1 - \vec{k}_4$. Using Equation 3.7 the continuation of a streamlines is calculated from multiple vectors instead of a single one. An illustration to help understand the method and compare both Euler and Runge-Kutta is shown in Figure 3.8.

Streamlines can either be generated extending from the seed point in the direction of flow only (prograde), or bidirectionally, extending in reverse flow direction (retrograde) as well. To extend a streamline in reverse flow direction, the vectors used for calculation are simply flipped by multiplying with -1.

Computing streamlines is a computationally expensive task, because a lot of calculations need to be done depending on the number, length and precision of the lines. For a typical set of streamlines, the line length, specified in number of segments, is 400 and the number of streamlines varies between 10 and 100.

For each segment of each line, Euler / Runge-Kutta calculations need to be performed separately. When generating streamlines bidirectionally, the amount of calculations is between 8000 and 80000. This may not seem to be a large number of computations, but with every segment calculation requiring the search of the nearest vector once in case of Euler and even four times for 4th order Runge-Kutta, the generation of a full set of streamlines can take minutes. Although our grid data structure drastically improves search times, better performance is desirable still.

By parallelizing the line generation process a huge performance gain can be achieved. Parallel calculation is performed on the CPU, computing lines simultaneously. This means, that the segments of one line are still being calculated in serial, while multiple lines can be build at once. When generating bidirectionally, the entire line is still calculated serially, by first building in flow direction until the desired amount of segments is reached and subsequently building in reverse flow direction. Splitting the prograde and retrograde building of a line can possibly further improve computation time, when enough individual computation units are available. However with a number of lines, that only in the least of cases falls below 20, the amount of cores available in commonly used CPUs is not large enough to improve performance significantly. Because the tests were performed on a machine with only 6 CPU-cores and 12 threads, streamline computation times where generally between 5 and 15 seconds, depending on number and length.

3.7 Streamline Visualization

After generating line structures based on hemodynamic data, the flow needs to be visualized by actually rendering the streamlines. First streamlines could simply be displayed using OpenGl line strip. With a list of 3D-locations, line strips display a chain of lines connecting al the given locations in the specified order. Rendering one line strip per streamline results in a basic visualization composed of lines, which are always rendered with a constant pixel width and color. This means, no shading and also no depth perception based on the width of a line was possible. While the generated line structures could be displayed using this method, more sophisticated rendering techniques are needed to achieve an effective visualization. A line representation including depth-dependent halos and an arrow glyph visualization were selected to be implemented, also incorporating animation, to increase perceptual effectiveness.

3.7.1 Depth-Dependent Halos

In order to produce an effective streamline visualization for a dense set of lines, we use depth-dependent halos, as presented by Everts et al. [4]. These halos create a border on either side of the streamline, which is colored the same as the background. This allows the viewer to assess the depth and distance of lines that cross or are close together. By using depth-dependent halos, the halo size is somewhat related to the distance between lines, appearing smaller for lines that are closer together and further away.

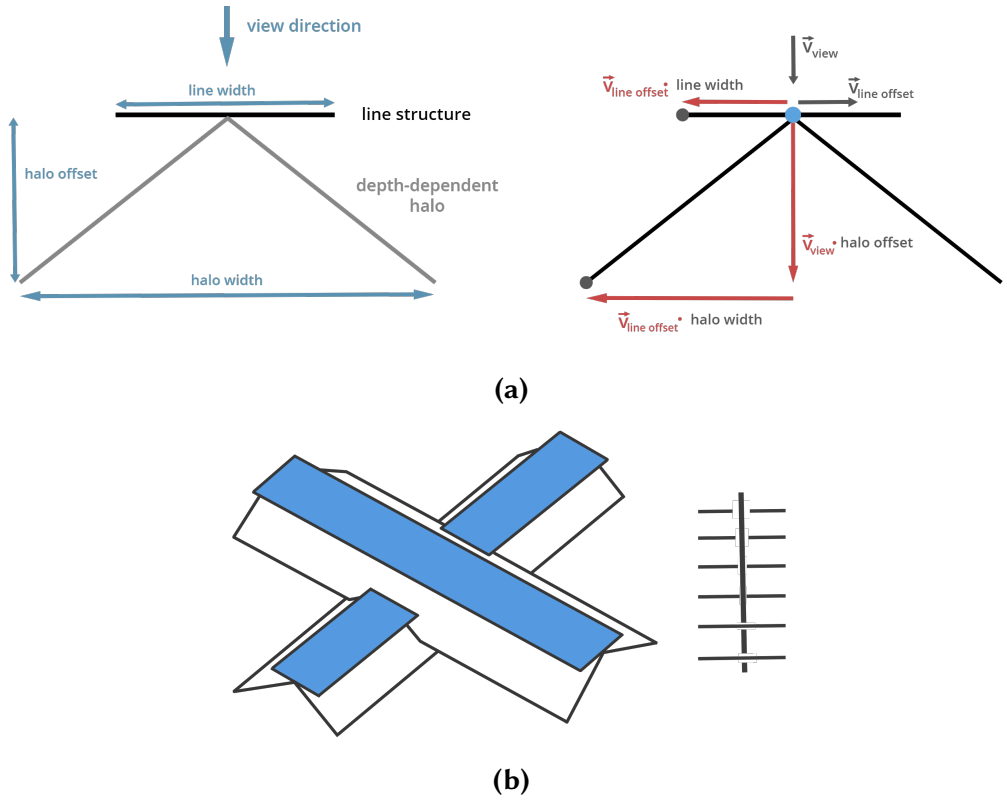


Figure 3.9: Illustration of depth-dependent halo assembly and effect, adapted from [4]. **(a)** Cross-sectional view of the line and halo and calculation of vertex locations required to build according meshes. **(b)** Illustration of the effect of depth dependent halos in 3D (left) and 2D (right). Lines running further away from the viewer are occluded, with the size of the halo dependent in the distance between the line at the crossing point.

Lines and halos are displayed as two separate meshes, always facing the viewer (the camera). With the line rendered as a set of faces perpendicular to the view direction, the halo extends laterally to the course of the streamline and back, further away from the camera.

Figure 3.9a shows an illustration of the cross-section of a line and its halo. It also presents an overview of how the vertex locations of line and halo mesh are acquired. The vector $\vec{v}_{\text{view}}$ is the view direction, calculated from the camera position and the position of the currently processed point $\vec{v}_p$ on the streamline and used to ensure all lines are facing the viewer. Perpendicular to both view direction and the vector between leading and trailing points on the streamline, $\vec{v}_{\text{line\_offset}}$ is used to build the line mesh. Placing vertices at $\vec{v}_p \pm \vec{v}_{\text{line\_offset}} \cdot \text{width}$, and connecting them into a mesh yields the desired line representation. To create the mesh for a halo, $\vec{v}_{\text{view}}$ and $\vec{v}_{\text{line\_offset}}$ are used together. The three vertices required to create a v-shape, you can see in Figure 3.9a, are generated as follows:

$$\begin{aligned} \vec{v}_{\text{left}} &= \vec{v}_p + \vec{v}_{\text{view}} \cdot \text{offset} + \vec{v}_{\text{line\_offset}} \cdot \text{width}, \\ \vec{v}_{\text{center}} &= \vec{v}_p, \\ \vec{v}_{\text{right}} &= \vec{v}_p + \vec{v}_{\text{view}} \cdot \text{offset} - \vec{v}_{\text{line\_offset}} \cdot \text{width}. \end{aligned} \tag{3.8}$$

Again, connecting vertices of adjacent points on the streamline a mesh is build, representing the halo. The interaction of two lines in close proximity crossing and the subsequent intersection of line and halo meshes is illustrated in Figure 3.9b. The figure also shows the effect from the viewers perspective. On the right side you can see a vertical line crossed by several horizontal lines. From top to bottom the horizontal lines get closer to the camera, switching from being behind, to being in front of the vertical line.

3.7.2 Velocity Color-Mapping

To display more information, color mapping on the lines can be used. The velocity of blood flow is an important measurement of hemodynamics and is critical to understanding complex flow characteristics. Therefore, the velocity of flow at each point on a streamline is indicated by color, intuitively including this information in the visualization. A simple gradient from red to white is used, red representing the maximum occurring velocity and white representing 0 mm/s. The velocity values are assigned to each point on the line structure during streamline generation.

3.7.3 Visualization of Retrograde Flow

In case of an aortic dissection, the false lumen can be pressurized during systole, especially if there does not exist an exit tear in the lower part of the aorta. In this case, the pressure built during systole releases back into the true lumen during diastole. This flow, opposite in direction compared to the flow during systole is called retrograde flow. Another phenomenon, also called retrograde flow, is the flow of blood through either true or false lumen, reverse to the main flow direction during systole. This can occur during systole or diastole and is mostly indicated by blood moving back up the aorta instead of down. Retrograde flow has gained interest in research lately and may be of clinical relevance. It can help assessment of dissections as well as identification of diseases like aortic valve stenosis. Therefore, a method for visualizing the flow direction of blood inside a vessel compared to the primary flow direction was implemented. Depending on the size of tear and results of the skeletonization process, this method is more applicable to the second definition of retrograde flow given in the previous paragraph. To visualize whether flow is prograde or retrograde, the direction of flow is compared to the overall course of the surrounding vessel. The flow direction is simply the vector between the two points of a streamline segment. Acquiring the primary flow direction at a specific line segment, on the other hand, requires additional processing. The centerline generated in the process described in Section 3.5.1, can be used as an approximation of the primary flow of a vessel. By calculating the vector between consecutive points of the centerline the direction of the vessel at a specific location can be determined.

In order to find the local direction, a simple search for the closest centerline vertex is performed. Afterwards the angle between the flow vector and the vector indicating the local direction of the vessel is calculated and assigned to every vertex of a streamline during streamline generation. If the angle is very small the flow is in line with the vessel, but if the angle exceeds 90°, the flow can be considered retrograde.

A visualization incorporating an indication for retrograde flow is produced similarly to the velocity color mapping. Instead of assigning a color relative to velocity, the color correlates to the angle between flow and vessel direction. We chose green for prograde flow and red for retrograde flow.

3.7.4 Arrow Glyphs

This visualization shows the line structures generated through integration using arrow glyphs. Arrows are placed on the line in constant segment intervals and change in length depending on the local velocity. The mesh making up the arrows is build similarly to the the line mesh described in Section 3.7.1, making the arrows always face the camera. The arrow mesh is build using the same $\vec{v}_{\text{line\_offset}}$ to create the adjustable width of the shaft of the arrow. Additionally, the variable *tip length* is used to specify the dimensions of the tip of the arrow. The length of the shaft depends on the length of the segment and consequently, the velocity of flow at the location of the arrow. The dimensions of the tip however do not change. While the color mapping techniques, aiming to increase the amount of information conveyed by the visualization are also applied, no halos are used in conjunction with arrow glyphs.

3.8 True and False Lumen Flow Visualization

When analyzing the flow characteristics of aortic dissections, comparing the flow of true and false lumen is important. Also recognizing the interaction of both flows at entry as well as exit tears and pressure in the lumina may help predict disease progression. Pressures are displayed on the surface of the vessel and dissection flap, as its the most relevant area. To effectively visualize the flow in both flow channels and enable direct comparison, streamlines generated separately for each lumen need to be displayed at the same time. The trimming performed to reduce artifacts while generating streamlines, described in Section 3.6.1, already creates separate volume grids for each lumen. With the vector field already separated, a set of streamlines for each lumen can be generated easily. Displaying both sets simultaneously however introduces clutter and occlusion. Also recognizing which lumen a specific line belongs to is difficult without actively differentiating the appearance of true and false lumen streamlines.

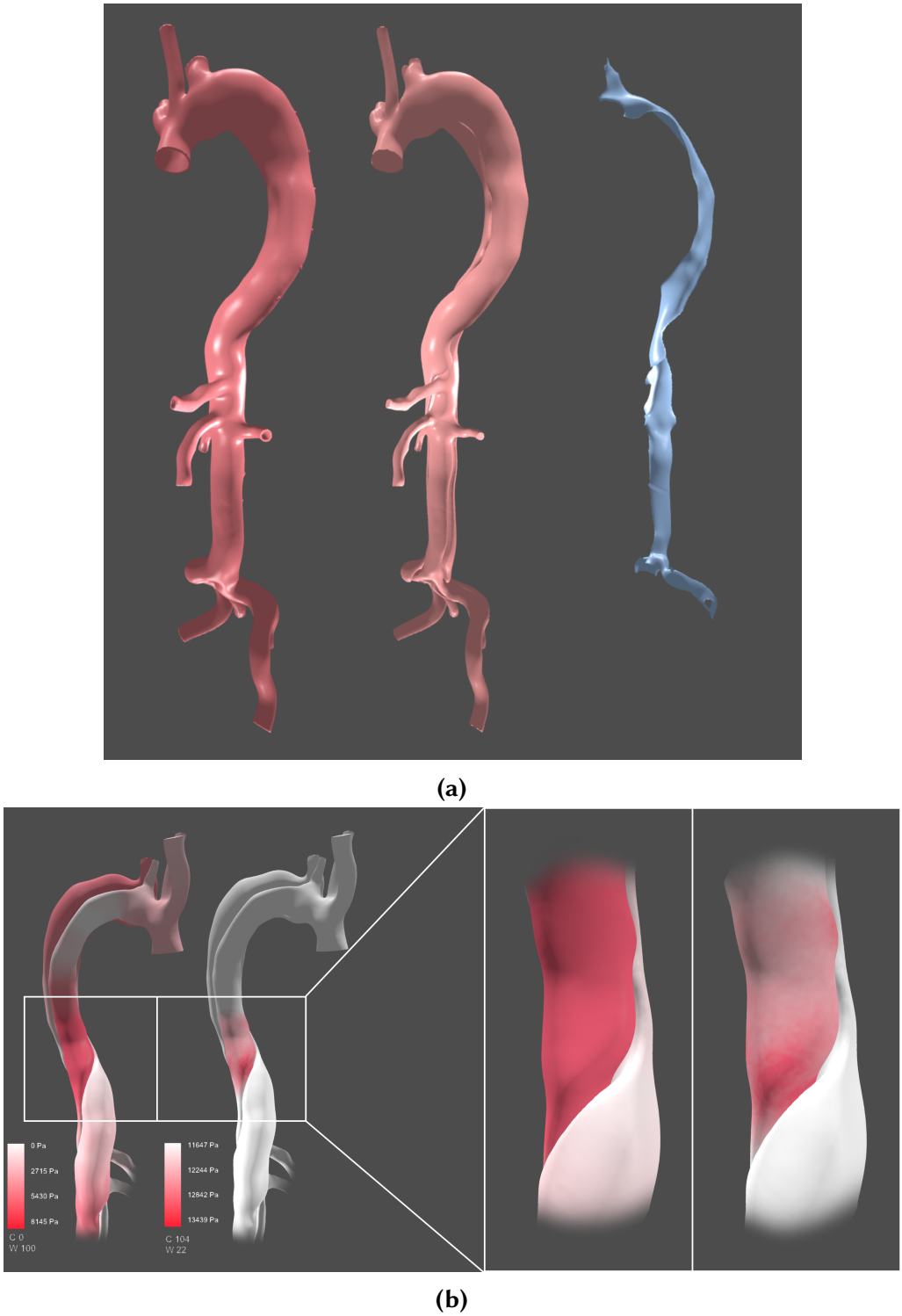
To differentiate two sets of streamlines, one set is displayed with a texture, similar to dashed lines, while the other one retains the previously described visualization. The dashed line visualization uses the same depth-dependent halo technique, augmented by color change in specific locations. On each segment of a streamline, half of the segment is darkened, producing a contrasting striped texture. This way, the original color mapping can still be used to convey more information, without the need of separate color scales for both lumina.

4. Results and Discussion

4.1 Surface Visualization

Visualization of medical datasets is mostly done in 2D using slices, not allowing for intuitive interpretation of spatial relations. Although direct volume rendering can produce 3D visualizations, they can still be difficult to understand. With the use of shaded mesh rendering, a more intuitive visualization can be created, allowing for better depth perception and spatial awareness, compared to direct volume rendering. The size, course and interaction of vessels and surfaces inside of the aorta can be recognized much more easily. Thanks to shading and specular reflection, the shape and curvature of surfaces, such as the dissection flap, is more intuitive to grasp. In Figure 4.1a, you can see the rendering of the outer surface of an aorta, the inner surface of the aorta (the surfaces in contact with blood) and the dissection flap. By separating the meshes and displaying them individually, surface features can be analyzed more easily, especially in case of the dissection flap.

Color mapping of pressure values on the surfaces of fluid and flap domain, clearly shows areas of elevated pressure, which may need to be investigated further. As the color mapping can be adjusted, the viewer can choose to analyze the whole aorta on a more macroscopic level or a specific area in detail. The effect of color mapping and color scale adjustment using the windowing function can be seen in Figure 4.1b. The left view clearly shows an area of elevated pressure in the true lumen, indicated by bright red color. Magnifying the view, the area of interest appears homogeneous, not allowing for precise localization of peak pressure areas. The windowing function is adjusted to map the color scale to a smaller portion of the pressure range. This results in the visualization shown in the right images. Generally, high pressure areas still appear mostly red, while smaller completely red areas can be identified as peak pressure zones.



(a)

(b)

Figure 4.1: Rendering of surface meshes extracted from the dataset and demonstration of color mapping including adjustment of color scale using windowing function. **(a)** Individual visualization of vessel wall (left), fluid/blood volume (middle) and dissection flap (right). **(b)** A color scale applied to the surface of the fluid volume indicating high and low pressure areas. The left view allows clear identification of a high pressure zone. By adjusting the windowing function the right view can be obtained and a more detailed analysis of this area can be performed.

Parameter	Range	Description
Seed Plane Position	0 - 100	Determines the start location for streamline generation.
Seed Point Spacing	0.4 - 5.0	Specifies the minimum distance between seed points.
Seed Point Margin	0.0 - 5.0	Specifies the minimum distance between seed points and vessel wall.
Line Segment Count	50 - 1000	Determines the length of generated streamlines.
Bidirectional Line Generation	true or false	Generate Streamlines down stream or in both directions, starting at the seed plane.
Line Width	0.2 - 2.0	Controls the thickness of lines.
Halo Width	0.0 - 5.0	Controlls relative width of depth-dependent halos.

Table 4.1: List of all the available parameters, with value range and a description of their effect.

4.2 Streamline Visualization

Streamlines visualize the flow inside of aortic dissections for a specific point in time. Although this is different to the pathlines usually used for the visualization of hemodynamics, because pressure and flow rate change over time, our results are comparable to other integral line structure visualizations of flow inside the aorta. As the goal of this work is to explore the applicability of different visualization techniques to aortic dissections, many parameters, including seed location, amount of streamlines, length, width (see Table 4.1) and color scaling, can be adjusted, producing various visualizations.

4.2.1 Seed Area Parameters

All available parameters, that alter the flow visualization are summarized in Table 4.1, with their respective value range and a short description. Images of the results as well as more comprehensive descriptions of every parameter, except bidirectional generation, are presented below.

The position of the seed plane can be adjusted to any point along the aortic centerline, that was extracted from the vessel tree. This allows for generation of shorter streamlines in a specified area of interest, which reduces computation time. Faster iterations and higher numbers of lines are possible without long waiting times, because it is not necessary to generate lines stretching the length of the entire aorta, only to analyze a small area. Along with selecting the starting location of the streamlines, the spacing and margin of seed points can be adjusted. While spacing mostly determines the amount of streamlines generated, the margin parameter can be used to only place seed points at a minimum distance from the wall of the flow channel. Both parameters are initially set to 2.0, resulting in a relatively small number of streamlines.

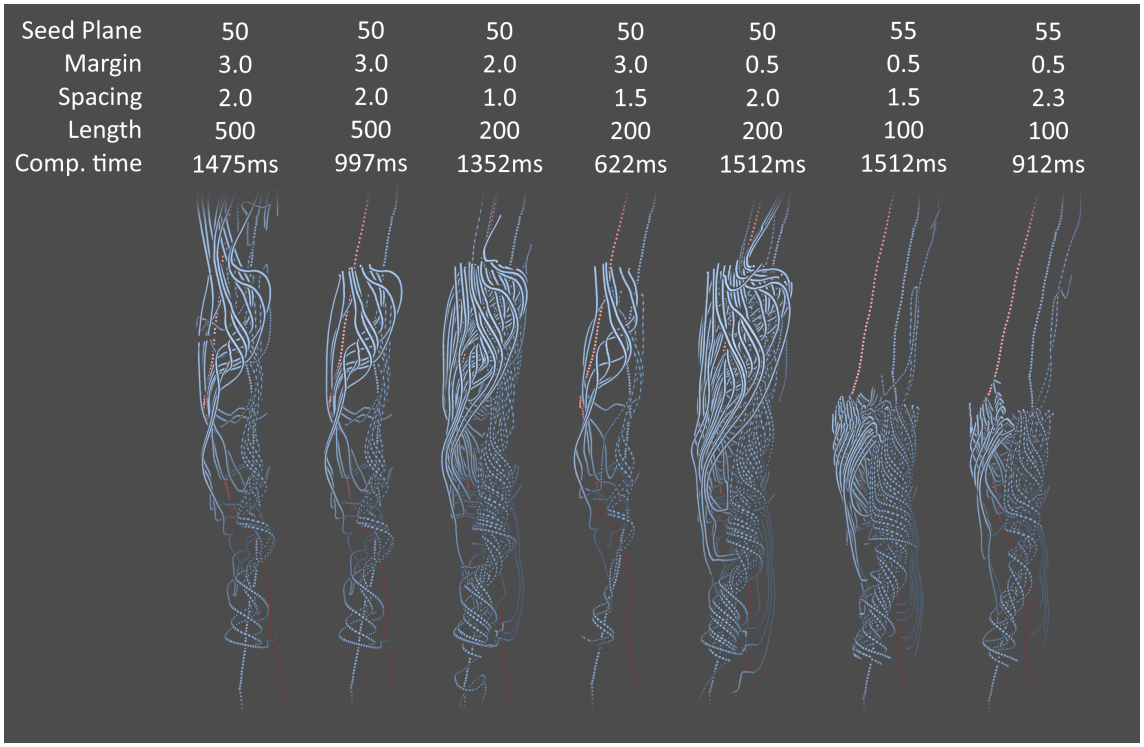


Figure 4.2: Effect of parameters (seed plane index, border margin, point spacing and line length) on the resulting streamlines and required time for generation. First and second example are generated with the same values for all parameters, except, the first example is generated in both directions, away from the seed plane, for 500 segments.

This helps fast iterations at the start to find areas of interest, after which parameters can be adjusted to reveal more detail. As previously mentioned, the length of the generated lines can be adjusted, simply stopping a line if it reaches a specified length.

Figure 4.2 shows the effect of parameters on the visualization result and the computation time. In these examples, the goal is to analyze the swirling flow inside the true lumen (striped lines). The first image is generated with streamlines extending in both direction starting at the seed plane. While this can be useful for filling an area of the vessel, when investigating the flow more generally, in this case half of the generated streamlines are not relevant for the analysis. In the remaining examples, the importance of choosing a seed location close to the investigated feature and choosing seed plane parameters to generate an appropriate set of streamlines is demonstrated. Choosing a very small value for spacing results in a large number of lines, possibly obstructing the view. The same problem can occur when choosing a low spacing in combination with low border margin, as small values results in more lines and consequently more occlusion and longer computation time.

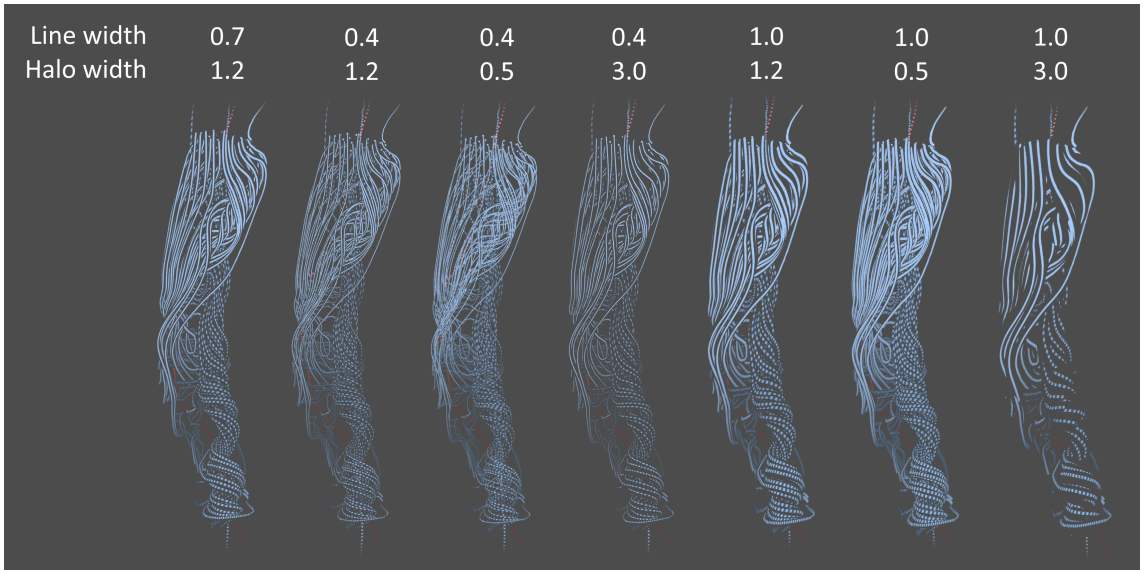


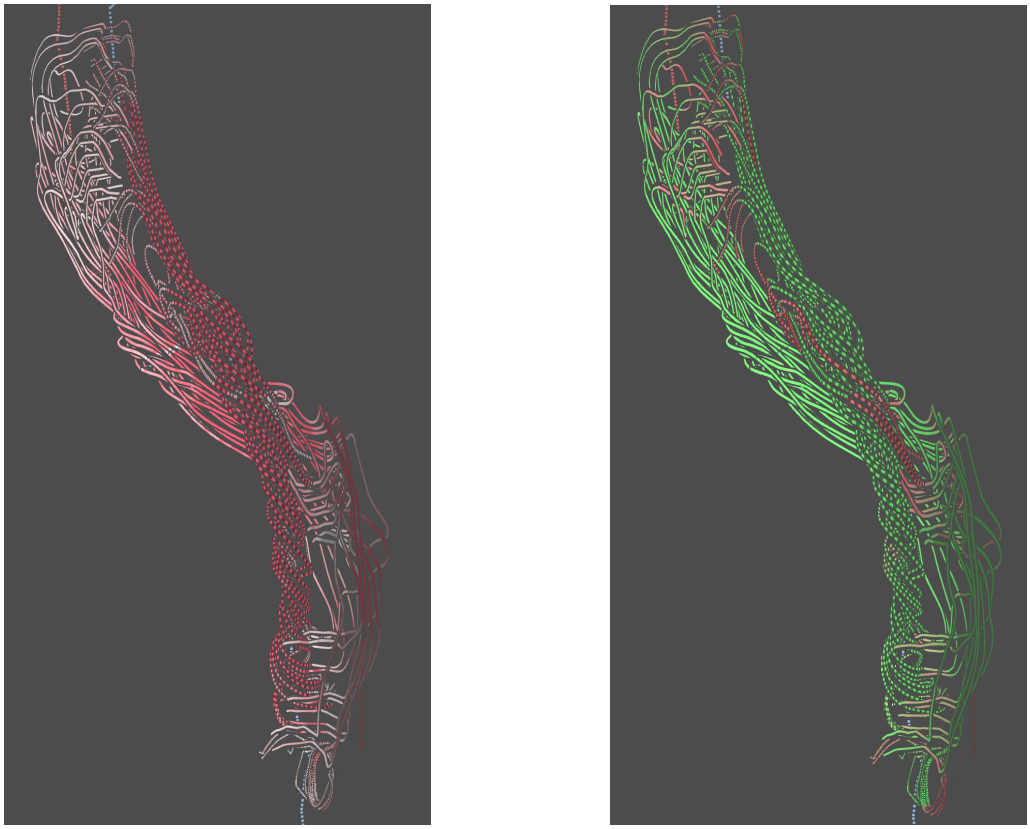
Figure 4.3: Example results of streamline rendering using depth dependent halos. Line width as well as halo width are adjusted to achieve different visualization goals.

Setting line length to a value that generates lines of reasonable length, that do not extend far beyond the feature being analyzed at the moment, can help decrease waiting times as well as reduce visual clutter. Comparing the first example using default values and the last example with the parameters adjusted to the visualization goal, the latter example displays the vortex flow more clearly, by generating lines in the desired area, while reducing computation time.

4.2.2 Depth-dependent Halos

Streamline visualizations often suffer from occlusion and visual clutter, caused by a large number of lines. The halos used to visualize streamlines help distinguish streamlines, reducing clutter. Directionality and depth of lines can be more easily recognized using this technique. Similar to the seed point generation, different parameters can be adjusted to alter the appearance of the streamlines. The parameters *line width* and *halo width* determine the thickness of lines and halos, while line width also adjusts the halo width proportionally, halo width can be used to alter the size of the halo independently.

To demonstrate the impact of changing those parameters, different examples are shown in Figure 4.3. For the first view, default values are applied, creating a visualization showing flow features in detail, while reducing clutter. In the next three images, the line width is reduced, producing thinner lines, reducing occlusion and therefore showing even more detail. Decreasing the halo width also reduces occlusion, but can lead to lines being hard to distinguish, if the halos become to small. This can be observed in the third image from the left.



(a) **(b)**

Figure 4.4: Color mapping techniques to increase information density of the visualization. True and false lumen are distinguished using texture, while simultaneously applying color mapping. **(a)** Streamlines colored according to velocity. Slow flow appears white, while fast flow is colored red. **(b)** Visualization of retrograde flow. Streamlines mostly parallel to the primary flow direction are colored green, reverse flow is colored red.

Increasing halo width, leads to more occlusion, but produces a less overloaded visualization, which may be suitable for more general assessment of flow. A similar effect can be achieved by increasing the line width, displayed in the last three images. Thicker lines and thicker halos minimize visual clutter, showing mostly streamlines that are close to the viewer.

4.2.3 Color-Mapping

To increase the amount of information shown by the visualization, lines are colored according to either velocity or the flow direction. Velocity color mapping allows for intuitive assessment of the flow rate in specific areas of the aorta. Faster and slower moving flow can be clearly differentiated, while the difference in coloring also helps distinguish individual lines. Streamlines belonging to different lumina are separated using a striped texture, allowing easy and clear identification of the belonging of a line. While textured lines generally appear slightly darker due to the dark stripes, high velocity is signaled by higher saturation. This allows direct comparison of velocities even while shading the lines, because the color scale does not depend on brightness.

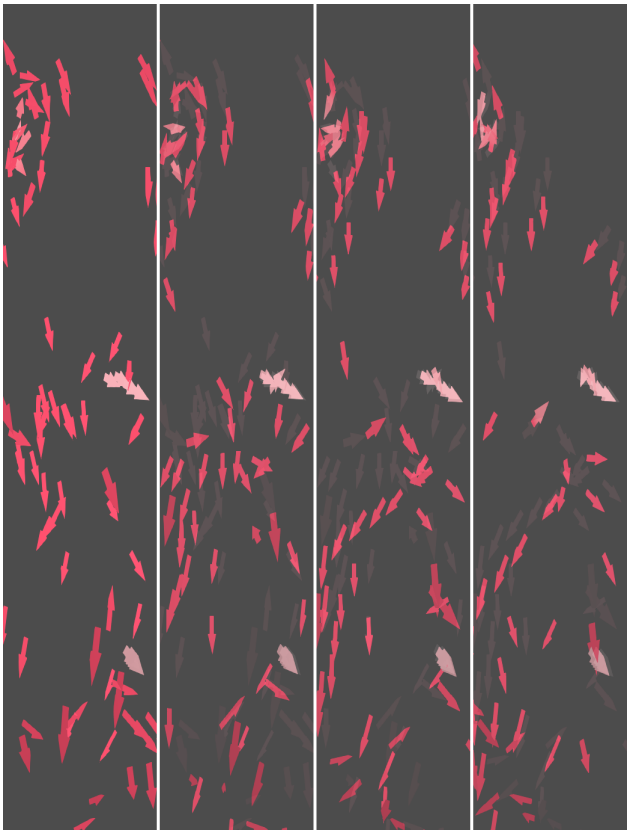


Figure 4.5: Example of animated arrow glyphs with red and white velocity color-coding. The animation progresses from left to right, with the previous frame overlayed with 17% opacity.

For the representation of flow direction in form of color mapping on the streamlines, a green to red gradient is used. Prograde flow, moving away from the heart is colored green, signaling regular flow. Retrograde flow however is colored red, alarming about irregular flow behavior. This visualization effectively colors areas of interest in red when analyzing flow direction, creating an intuitive visualization, that draws the viewers attention towards retrograde flow features. Flow moving opposite to the natural flow direction of the vessel is clearly marked, while flow mostly perpendicular to the primary flow direction appears neither bright green nor bright red. Therefor retrograde flow occurring at fenestrations is not clearly marked, but could be discovered by investigating areas colored in a lighter yellowish color.

4.3 Arrow Glyph Animation

The direction of flow can be indicated effectively using arrows. However displaying too many arrow glyphs at once leads to high amounts of clutter, making it difficult to extract information from a visualization. When the number of arrows glyphs is reduced, visual clutter is reduced as well, but flow paths can not be recognized anymore, because the gaps between arrows increase in size. Using animation, a smaller number of arrows can be displayed, while flow paths can still be recognized. As arrows glyphs are built from streamline segments, arrow length corresponds to velocity of flow.

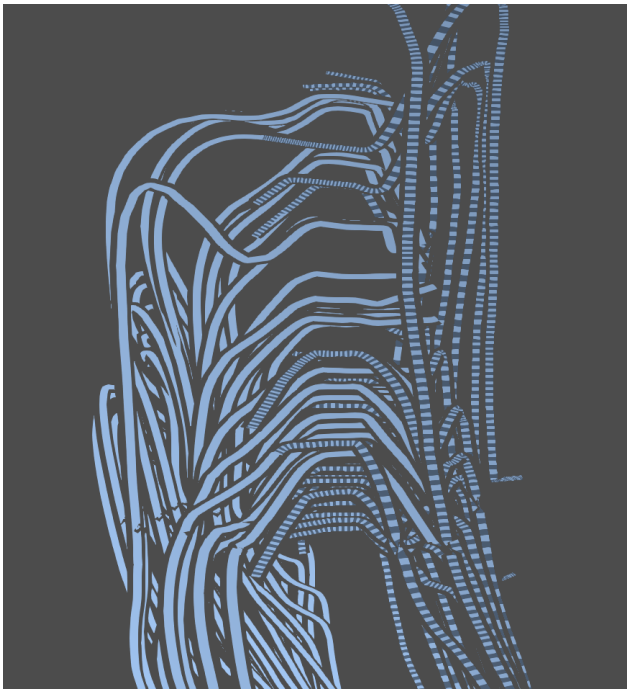


Figure 4.6: Streamlines falsely entering the dissection flap, causing them to run perpendicular to the surface of the flap. Without lumen specific trimming, streamlines would enter another lumen and continue instead of stopping inside the dissection flap.

Also caused by the way arrows glyphs are produced, arrows passing through high velocity flow, move faster in the animation. The movement of arrow glyphs along the streamline, they are placed on, produces a very intuitive visualization, similar to how flow can be observed in practice. This way flow can be intuitively visualized with relatively low clutter, depending on the number of arrows and animation speed.

For the animated arrow glyph visualization, the same color mapping techniques used on streamlines are applied. Velocity color mapping simplifies identification of particularly fast or slow flow, as differences in arrow length are not as striking as significant differences in color. Also, while the velocity of flow can be intuitively identified during animation, color helps to speed up this process. Color mapping of retrograde flow does not improve the identification of retrograde flow very much, as single arrows glyphs moving in opposite direction compared to the majority of arrow glyphs are easily identified.

4.4 Limitations

Generating long continuous streamlines filling the entire aorta was impossible using the algorithms used in this work, and without additional mechanisms. This is caused by the majority of streamlines eventually entered a section of flow perpendicular to the vessel wall. Streamlines would therefore advance into the vessel wall and dissection flap and continue in the other lumen, suggesting the occurrence of cross lumen flow and consequently fenestrations. In areas where this phenomenon occurred, no fenestrations were present and the visualization therefor wrong.

This behavior is caused by the fact, that the CFD simulation used deformable models. In the simulation, the dissection flap moves drasticly, as the pressure in true and false lumina change. Because blood is not compressible, the fluid adjacent to the moving tissue has to follow the movement, creating flow close to perpendicular to the dissection flap or vessel wall. In order to mitigate the problem, trimming was performed, limiting streamlines of a particular lumen to only advance in the bound of that lumen. This prevents lines from passing through the dissection flap and continuing in another lumen, but does not prevent them from entering the vessel wall, as shown in Figure 4.6.

Another limitation of this work is, that retrograde flow is only detected by comparing flow direction to the ideal laminar flow through a vessel, using the centerline. Retrograde flow occurring inside a single lumen is detected and visualized, but retrograde flow occurring through fenestrations cannot be detected.

5. Conclusion and Future Work

While exploring flow visualization techniques for aortic dissections, streamlines were tested in combination with several methods to increase information density and perceptual effectiveness. Streamlines are well known and commonly used, but can produce occluded and cluttered visualizations. Through the use of depth-dependent halos, visualizations are less cluttered and easier to grasp. Color mapping velocity is also widely used and applicable to the visualization of flow in aortic dissections. As the clinical importance of retrograde flow is researched more, color mapping flow direction may be a useful tool in prognosis and treatment of dissections. Differentiated visualization of flow associated with true and false lumina allows for direct comparison of both flow channels. Possible interaction of flow at fenestrations may be of interest, but was not tested during this work. Animation of arrow glyphs also shows to be an effective visualization of flow in aortic dissections, but could be improved by using techniques, like the one proposed by Jobard et al. [37]. The dynamic representation, while not as accurate, produces a very intuitive visualization, able to simultaneously show flow paths, velocity and direction. A model of the aorta needs to be shown to be able to assess the location of specify flow features. While color mapping pressure values is not flow visualization, analysis of pressure inside the vessel is an important part of disease prognosis and treatment.

The simulations provide flow data covering the entire cardiac cycle. For this work however, only the data of a single time point was analyzed. Pathlines, streaklines or timelines may be generated with the use of multiple datasets, to analyze flow over time. As streamlines only show instantaneous flow, pathlines may be more accurate to real life hemodynamics. This could also mitigate the problem of lines passing through tissue. Incorporating multiple times steps in the cardiac cycle, a more accurate analysis of retrograde flow may be possible, by comparing flow at several locations at different points in time. A better method for selection of true lumen, false lumen and joined aorta could lead to effective visualizations for the analysis of true and false lumen flow interaction.

Bibliography

[1] Bradley Allen, Pascale Aouad, Nicholas Burris, Amir Ali Rahsepar, Kelly Jarvis, Christopher Francois, Alex Barker, Sukit Malaisrie, James Carr, Jeremy Collins, and Michael Markl. Detection and Hemodynamic Evaluation of Flap Fenestrations in Type B Aortic Dissection with 4D Flow MRI: Comparison with Conventional MRI and CT Angiography. *Radiology: Cardiothoracic Imaging*, 1(1):1–7, April 2019. (cited on Page xiii, 5, and 6)

[2] M. Gharib, Dymphy Kremers, M. Koochesfahani, and Martin Kemp. Leonardo's vision of flow visualization. *Experiments in Fluids*, 33(1):219–23, January 2002. (cited on Page xiii and 7)

[3] Robert Kirby, Haralambos Marmanis, and David Laidlaw. Visualizing Multivalued Data from 2D Incompressible Flows Using Concepts from Painting. In *Proceedings of IEEE Visualization*, pages 333–40, October 1999. (cited on Page xiii and 8)

[4] Maarten H. Everts, Henk Bekker, Jos B. T. M. Roerdink, and Tobias Isenberg. Depth-Dependent Halos: Illustrative Rendering of Dense Line Data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299—-306, November 2009. (cited on Page xiii, 24, and 25)

[5] Maximilian Wundram, Volkmar Falk, Jaime-Jürgen Eulert-Grehn, Hermann Herbst, Jana Thurau, Bernd A. Leidel, Eva Göncz, Wolfgang Bauer, Helmut Habazettl, and Stephan Kurz. Incidence of acute type A aortic dissection in emergency departments. *Scientific Reports*, 10(1):7434, May 2020. (cited on Page 1)

[6] Kathrin Bäumler, Vijay Vedula, Anna Sailer, Jongmin Seo, Peter Chiu, Gabriel Mistelbauer, Frandics Chan, Michael Fischbein, Alison Marsden, and Dominik Fleischmann. Fluid–structure interaction simulations of patient-specific aortic dissection. *Biomechanics and Modeling in Mechanobiology*, 19(5):1607–28, October 2020. (cited on Page 1, 2, 3, 5, and 11)

[7] Ravi Hebballi and J. Swanevelder. Diagnosis and management of aortic dissection. *Continuing Education in Anaesthesia, Critical Care & Pain*, 9(1):14–18, February 2009. (cited on Page 1)

[8] Domenico Spinelli, Filippo Benedetto, Rocco Donato, Gabriele Piffaretti, Massimiliano Marrocco-Trischitta, Himanshu Patel, Kim Eagle, and Santi Trimarchi.

Current evidence in predictors of aortic growth and events in acute type B aortic dissection. *Journal of Vascular Surgery*, 68(6):1925–35, August 2018. (cited on Page 1)

[9] Jin Wook Chung, Christopher Elkins, Toyohiko Sakai, Noriyuki Kato, Thomas Vestring, Charles Semba, Suzanne Slonim, and Michael Dake. True-lumen collapse in aortic dissection: part I. Evaluation of causative factors in phantoms with pulsatile flow. *Radiology*, 214(1):87–98, February 2000. (cited on Page 1)

[10] Jin Wook Chung, Christopher Elkins, Toyohiko Sakai, Noriyuki Kato, Thomas Vestring, Charles P. Semba, Suzanne M. Slonim, and Michael D. Dake. True-lumen collapse in aortic dissection: part II. Evaluation of treatment methods in phantoms with pulsatile flow. *Radiology*, 214(1):99–106, 2000. (cited on Page 1)

[11] Ramon Berguer, Juan Parodi, Marty Schlicht, and Khalil Khanafer. Experimental and Clinical Evidence Supporting Septectomy in the Primary Treatment of Acute Type B Thoracic Aortic Dissection. *Annals of Vascular Surgery*, 29(2):167–73, October 2015. (cited on Page 1)

[12] Thomas T. Tsai, Marty S. Schlicht, Khalil Khanafer, Joseph L. Bull, Doug T. Valassis, David M. Williams, Ramon Berguer, and Kim A. Eagle. Tear size and location impacts false lumen pressure in an ex vivo model of chronic type B aortic dissection. *Journal of Vascular Surgery*, 47(4):844–51, April 2008. (cited on Page 1)

[13] Zhuo Cheng, F. Tan, C. Riga, Colin Bicknell, Mohamad Hamady, Richard Gibbs, Nigel Wood, and Xiao Xu. Analysis of flow patterns in a patient-specific aortic dissection model. *Journal of Biomechanical Engineering*, 132(5):051007, May 2010. (cited on Page 1)

[14] Barry Doyle and Paul Norman. Computational biomechanics in thoracic aortic Dissection: Today's Approaches and Tomorrow's Opportunities. *Annals of Biomedical Engineering*, 44(1):71–83, June 2015. (cited on Page 1)

[15] Anna Sailer, Sander Kuijk, Patricia Nelemans, Anne Chin, Aya Kino, Mark Huininga, Johanna Schmidt, Gabriel Mistelbauer, Kathrin Bäumler, Peter Chiu, Michael Fischbein, Michael Dake, D. Miller, Geert Schurink, and Dominik Fleischmann. Computed Tomography Imaging Features in Acute Uncomplicated Stanford Type-B Aortic Dissection Predict Late Adverse EventsCLINICAL PERSPECTIVE. *Circulation: Cardiovascular Imaging*, 10(4):e005709, April 2017. (cited on Page 1)

[16] Anna Sailer, Patricia Nelemans, Trevor Hastie, Anne Chin, Mark Huininga, Peter Chiu, Michael Fischbein, Michael Dake, D. Miller, Geert Schurink, and Dominik Fleischmann. Prognostic Significance of Early Aortic Remodeling in Acute Uncomplicated Type B Aortic Dissection and Intramural Hematoma. *The Journal of Thoracic and Cardiovascular Surgery*, 154(4):1192–200, October 2017. (cited on Page 1)

[17] Alison Marsden. Simulation based planning of surgical interventions in pediatric cardiology. *Physics of Fluids*, 25(10):101303, October 2013. (cited on Page 1)

[18] Charles Taylor and David Steinman. Image-Based Modeling of Blood Flow and Vessel Wall Dynamics: Applications, Methods and Future Directions. *Annals of Biomedical Engineering*, 38(3):1188–203, March 2010. (cited on Page 1)

[19] Christopher J. François, Michael Markl, Mark L. Schiebler, Eric Niespodzany, Benjamin R. Landgraf, Christian Schlensak, and Alex Frydrychowicz. Four-dimensional, flow-sensitive magnetic resonance imaging of blood flow patterns in thoracic aortic dissections. *The Journal of Thoracic and Cardiovascular Surgery*, 145(5):1359–66, May 2013. (cited on Page 2)

[20] Christoph Nienaber, Hervé Rousseau, Holger Eggebrecht, Stephan Kische, Rossella Fattori, Tim Rehders, Günther Kundt, Dierk Scheinert, Martin Czerny, Tilo Kleinfeldt, Burkhart Zipfel, Louis Labrousse, and Hüseyin Ince. Randomized Comparison of Strategies for Type B Aortic Dissection: The INvestigation of STEnt Grafts in Aortic Dissection (INSTEAD) Trial. *Circulation*, 120(25):2519–28, December 2009. (cited on Page 2)

[21] Gebre Tefera, Charles Acher, John Hoch, Mathew Mell, and William Turnipseed. Effectiveness of intensive medical therapy in type B aortic dissection: A single-center experience. *Journal of Vascular Surgery*, 45(6):1114–8, July 2007. (cited on Page 2)

[22] Rachel E. Clough, Matthew Waltham, Daniel Giese, Peter R. Taylor, and Tobias Schaeffter. A new imaging method for assessment of aortic dissection using four-dimensional phase contrast magnetic resonance imaging. *Journal of Vascular Surgery*, 55(4):914–23, 2012. (cited on Page 2)

[23] Eric K. Shang, Derek P. Nathan, Ronald M. Fairman, Joseph E. Bavaria, Robert C. Gorman, Joseph H. Gorman, and Benjamin M. Jackson. Use of computational fluid dynamics studies in predicting aneurysmal degeneration of acute type B aortic dissections. *Journal of Vascular Surgery*, 62(2):279–84, 2015. (cited on Page 2)

[24] Anja Oßwald, Christof Karmonik, Jeff Anderson, F. Rengier, M. Karck, J. Engelke, Klaus Kallenbach, Drosos Kotelis, Sasan Partovi, D. Böckler, and Arjang Ruhparwar. Elevated Wall Shear Stress in Aortic Type B Dissection May Relate to Retrograde Aortic Type A Dissection: A Computational Fluid Dynamics Pilot Study. *European Journal of Vascular and Endovascular Surgery*, 54(3):324–30, July 2017. (cited on Page 2)

[25] Zhuo Cheng, Nigel Wood, Richard Gibbs, and Xiao Xu. Geometric and Flow Features of Type B Aortic Dissection: Initial Findings and Comparison of Medically Treated and Stented Cases. *Annals of Biomedical Engineering*, 43(1):177–89, May 2014. (cited on Page 2)

[26] C. Karmonik, J. Bismuth, D. Shah, M. Davies, D. Purdy, and A. Lumsden. Computational study of haemodynamic effects of entry- and exit-tear coverage in a DeBakey type III aortic dissection: technical report. *European Journal of Vascular and Endovascular Surgery*, 42 2(2):172–7, August 2011. (cited on Page 2)

[27] Simon Yu, Wen Liu, Randolph Wong, Malcolm Underwood, and Defeng Wang. The Potential of Computational Fluid Dynamics Simulation on Serial Monitoring of Hemodynamic Change in Type B Aortic Dissection. *Cardiovascular and Interventional Radiology*, 39(8):1090–8, August 2016. (cited on Page 2)

[28] D. Dillon-Murphy, A. Noorani, D. Nordsletten, and C. A. Figueroa. Multi-modality image-based computational analysis of haemodynamics in aortic dissection. *Biomechanics and Modeling in Mechanobiology*, 15(4):857–76, 2016. (cited on Page 2)

[29] Zhuo Cheng, C. Juli, N. Wood, R. Gibbs, and X. Xu. Predicting flow in aortic dissection: Comparison of computational model with PC-MRI velocity measurements. *Medical Engineering & Physics*, 36(9):1176–84, 2014. (cited on Page 2)

[30] Mirko Bonfanti, Stavroula Balabani, John Greenwood, Sapna Puppala, Shervanthi Homer-Vanniasinkam, and Vanessa Díaz-Zuccarini. Computational tools for clinical support: A multi-scale compliant model for haemodynamic simulations in an aortic dissection based on multi-modal imaging data. *Journal of The Royal Society Interface*, 14(136):20170632, November 2017. (cited on Page 2)

[31] Timothy Urness, Victoria Interrante, I. Marusic, Ellen Longmire, and Bharathram Ganapathisubramani. Effectively Visualizing Multi-Valued Flow Data using Color and Texture. In *IEEE Visualization*, pages 115–21. IEEE, November 2003. (cited on Page 3)

[32] Benjamin Köhler, Matthias Grothoff, Matthias Gutberlet, and Bernhard Preim. Pressure-based vortex extraction in cardiac 4D PC-MRI blood flow data. In *EuroVis 2018 - Short Papers*, pages 13–17, June 2018. (cited on Page 6)

[33] Benjamin Köhler, Matthias Grothoff, Matthias Gutberlet, and Bernhard Preim. Visual and quantitative analysis of blood flow jets in cardiac 4D PC-MRI data. *Computer Graphics Forum*, 37(3):195–204, May 2018. (cited on Page 6)

[34] Andrea Brambilla, Robert Carnecky, Ronald Peikert, Ivan Viola, and Helwig Hauser. Illustrative Flow Visualization: State of the Art, Trends and Challenges. In *Eurographics 2012 - State of the Art Reports*, pages 75–94, 2012. (cited on Page 6 and 8)

[35] Benjamin Behrendt, Wito Engelke, Philipp Berg, Oliver Beuing, Bernhard Preim, Ingrid Hotz, and Sylvia Saalfeld. Evolutionary Pathlines for Blood Flow Exploration in Cerebral Aneurysms. In *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 253–63, 2019. (cited on Page 8)

[36] Bernhard Preim and Monique Meuschke. A Survey of Medical Animations. *Computers Graphics*, 90(1):145–68, August 2020. (cited on Page 9)

[37] B. Jobard, Nicolas Ray, and D. Sokolov. Visualizing 2D Flows with Animated Arrow Plots. *ArXiv*, 1205.5204, 2012. (cited on Page 39)